MASTER EN SCIENCES INFORMATIQUES

Développement d'un prototype de système expert pour la planification d'expériences en biochimie

Lebrun, Y.; Lhermitte, B.

Award date:
1988

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix
Institut d'Informatique

DEVELOPPEMENT D'UN PROTOTYPE DE SYSTEME EXPERT POUR LA PLANIFICATION D'EXPERIENCES EN BIOCHIMIE

Y. LEBRUN et B. LHERMITTE

Mémoire présenté en vue de l'obtention du titre de licencié et maître en Informatique

Promoteur : M. NOIRHOMME-FRAITURE
Directeur : J. BARRETO

Septembre 1988

Facultés Universitaires Notre-Dame de la Paix
Institut d'Informatique
rue de Bruxelles, 61, B-5000 **NAMUR**

DEVELOPPEMENT D'UN PROTOTYPE DE SYSTEME EXPERT POUR LA PLANIFICATION D'EXPERIENCES EN BIOCHIMIE

Y. LEBRUN et **B. LHERMITTE**

Résumé
Ce mémoire développe la conception d'un prototype de système expert destiné à la planification d'expériences en biochimie. Il est le fruit d'une collaboration avec le laboratoire de biochimie cellulaire de la Faculté des Sciences de Namur. Les expériences concernent la purification de substances d'origine biologique extraites de tissus cellulaires. Nous avons considéré les trois techniques de purification suivantes : la chromatographie sur tamis moléculaire, la chromatographie sur échangeur d'ions et la précipitation isoélectrique. Les systèmes de production ont été utilisés comme technique de représentation des connaissances. L'implémentation du prototype a été faite en LISP sur MS-DOS. Le moteur d'inférence implémenté met en oeuvre un mécanisme de "pattern-matching" étendu et une stratégie de résolution en chaînage avant.

Abstract
This dissertation presents the construction of an expert system prototype for the planification of biochemical experiments. It results from a collaboration with the cellular biochemistry laboratory of the Science School in Namur. The experiments concern the isolation of biochemical substances. Three techniques were faced : gel filtration, ion exchange chromatography and isoelectric precipitation. Production systems were used to represent the knowledge. The tool for the prototype implementation was LISP. The inference engine applies extented pattern matching and forward chaining strategy.

Mémoire de licence et maîtrise en Informatique
Septembre 1988
Promoteur : **M. NOIRHOMME-FRAITURE**
Directeur : **J. BARRETO**

Remerciements

Bien plus qu'un devoir, c'est un plaisir pour nous de remercier Monsieur Jorge Barreto, directeur de ce mémoire. Sa grande disponibilité et ses précieux conseils nous ont beaucoup aidé dans la réalisation de ce travail. Nous remercions également Madame Monique Noirhomme qui a accepté d'être notre promoteur.

Nous tenons à exprimer notre reconnaissance, pour son accueil, à tout le personnel du laboratoire de biochimie cellulaire de la Faculté des Sciences. Que soient spécialement remerciés Monsieur José Remacle, directeur de ce laboratoire, pour le temps qu'il a bien voulu consacrer à répondre à nos nombreuses questions, ainsi que Messieurs Edouard Delaive et Marc Roger.

Nous exprimons enfin notre gratitude à toutes les personnes qui, de près ou de loin, ont contribué à la réalisation de ce mémoire.

TABLE DES MATIERES

INTRODUCTION

Les systèmes experts ou "knowledge-based" systèmes sont des applications pratiques de recherches menées en intelligence artificielle. Leur développement consiste à essayer d'allier la puissance des ordinateurs et la richesse de l'expertise humaine. L'expertise est un ensemble de connaissances approfondies d'un domaine particulier, une compréhension des problèmes qui se posent dans ce domaine et les capacités de résoudre certains de ces problèmes.

Les systèmes experts sont conçus pour représenter et appliquer les connaissances d'un domaine étroit de l'expertise humaine pour résoudre au mieux des problèmes posés dans ce domaine. Par exemple, des efforts conjugués d'experts humains et de concepteurs de systèmes experts, souvent appelés ingénieurs de connaissances, ont permis le développement de systèmes prototypes dans des domaines comme le diagnostic médical, la configuration d'ordinateurs ou encore, la prospection minière. Les performances de certains systèmes sont élevées, proches de la performance des experts humains.

Des efforts sont maintenant réalisés pour exploiter cette technologie et l'étendre à de nouvelles applications où l'expertise humaine est requise. Avec le temps, les systèmes experts peuvent avoir un impact sur tous les domaines de l'activité humaine où la connaissance est essentielle pour résoudre les problèmes importants. Un des avantages de cette expertise "artificielle" est qu'elle est permanente. Les systèmes experts peuvent également contribuer à clarifier et à développer la connaissance humaine elle-même.

Un ensemble de principes, de techniques et d'outils ont été développés pour permettre le développement de systèmes experts. Les concepteurs tentent d'utiliser au mieux ces différents éléments. Cependant, les capacités d'un système expert viennent des connaissances qu'il incorpore plutôt que de formalismes et de schémas d'inférence particuliers : les connaissances de l'expert humain sont la clé de la performance du système expert, tandis que la représentation des connaissances et les schémas d'inférence fournissent des mécanismes d'utilisation des connaissances.

L'objectif de ce mémoire est de développer un premier prototype pour aider à la planification d'expériences en biochimie.

De nombreuses applications de type "système expert" ont abordé des domaines comme la médecine et le domaine militaire. Des disciplines scientifiques comme la physique, la chimie, la géologie ont également fait l'objet de développements de systèmes experts. Par contre, le domaine de la biologie a encore été peu abordé jusqu'ici. La raison n'est certainement pas que l'approche "système expert" est inadéquate pour cette discipline. Les systèmes experts sont adéquats pour résoudre, entre autres, des tâches d'interprétation et de planification comme il en existe en biologie. Les possibilités qu'ont acquises les systèmes experts à l'heure actuelle ont donné au directeur de notre mémoire l'idée de créer un système qui pourrait aider les biologistes lorsqu'ils réalisent des expériences. C'est ainsi que des contacts ont été pris avec le laboratoire de biochimie cellulaire de la Faculté des Sciences.

Notre rôle consistait dans un premier temps, à observer les différents domaines étudiés au laboratoire et à découvrir un problème particulier qu'il serait intéressant de traiter selon une approche "système expert". Ensuite,

nous souhaitions prendre connaissance du problème choisi et développer un prototype qui permettrait de montrer l'intérêt du développement d'un véritable système expert.

Nous souhaitions, parallèlement, à cette recherche, approfondir les connaissances de base acquises dans le cours "Techniques d'Intelligence Artificielle" de deuxième licence. Le développement d'un programme permet de réellement "toucher" les problèmes posés au concepteur d'une application dans le domaine des systèmes experts.

Notre travail est découpé en trois parties.

Dans une première partie, intitulée "Cadre général", nous présentons une première approche à la fois du domaine de la biochimie et des systèmes experts. Dans un premier chapitre, nous décrivons les démarches réalisées au laboratoire pour découvrir un problème adéquat pour le développement d'un système expert. Le deuxième chapitre est consacré à la présentation du domaine de la purification des substances d'origine biologique. Nous présentons l'intérêt que représente ce type d'expériences en biochimie, les problèmes posés dans le domaine et la manière habituellement employée pour les résoudre, et les techniques de purification auxquelles nous nous sommes intéressés. Ce chapitre est particulièrement destiné à la compréhension du domaine par les informaticiens. Le troisième chapitre présente les systèmes experts aux non-spécialistes et justifie pourquoi il nous a semblé adéquat de proposer un programme de type "système expert" pour aider à la purification de substances d'origine biologique.

La deuxième partie de notre travail est intitulée "Cadre méthodologique". Elles est constituée de deux chapitres. Le premier traite d'une méthode pour la construction d'un système expert. Le deuxième chapitre de

cette partie est destiné à présenter différents outils qui ont été développés par les chercheurs en intelligence artificielle pour aider les ingénieurs de connaissances lors de la conception de systèmes experts.

La troisième partie comprend les chapitres qui illustrent les différentes étapes que nous avons suivies pour concevoir le prototype.

Enfin, une évaluation de notre travail est proposée en fonction des objectifs que nous nous étions fixés au départ. Nous présentons également différents travaux qui pourraient constituer une suite à notre travail.

On trouvera en annexe le code du prototype.

PARTIE 1

CADRE GENERAL

CHAPITRE 1. UNE PREMIERE APPROCHE DU DOMAINE DE LA BIOCHIMIE

Nous allons, dans ce chapitre, présenter le cheminement qui nous a permis d'arriver à une collaboration avec le laboratoire de biochimie cellulaire.

1.1. POINT DE DEPART D'UNE COLLABORATION

Notre stage au laboratoire de biochimie cellulaire de la Faculté des Sciences constituait une première collaboration entre l'Institut d'Informatique et ce laboratoire dans le cadre d'un mémoire. Il a semblé au directeur de notre mémoire et au responsable de ce laboratoire qu'une telle collaboration pouvait présenter des intérêts de part et d'autre.

Du côté du laboratoire, le travail n'est actuellement possible que grâce à l'utilisation d'une technologie de pointe qui fait souvent appel à l'informatique. Les biochimistes sont confrontés à l'informatique et ils sont intéressés par tout ce qui est susceptible de leur permettre de se familiariser avec le domaine.

Pour notre part, nous envisagions la possibilité de développer un outil informatique ayant la capacité d'aider les biochimistes dans leur travail. Nous souhaitions dans un premier temps observer leurs recherches et leurs expériences, pour essayer de découvrir des possibilités de développement auxquelles ils ne pouvaient penser par manque de temps et de connaissances dans le domaine informatique.

Nous cherchions plutôt une possibilité de travail sur une tâche abordable par une approche "intelligence

artificielle" et plus spécifiquement "système expert". Nous pensions à la possibilité de développer un système d'aide à l'interprétation d'expériences ou à la conception d'expériences.

L'optique de notre stage était d'établir les premiers contacts avec le laboratoire. Il a principalement consisté en un travail de prospection et de familiarisation avec les activités des biochimistes, nos connaissances en biologie et en chimie étant au départ très réduites.

1.2. LA BIOLUMINESCENCE

Les différentes activités du laboratoire sont les suivantes :

- étude des modifications apparaissant dans les cellules au cours de leur vieillissement en culture
- étude des relations entre les cellules et les molécules qui interviennent dans les processus inflammatoires et allergiques (en collaboration avec l'industrie pharmaceutique)
- étude de la stabilisation thermique des enzymes immobilisés et application à la mise au point de bio-réacteurs de deuxième génération
- mise au point de nouveaux procédés de dosage en bioluminescence.

La bioluminescence est une réaction biologique d'une substance qui se traduit par le dégagement de photons. A notre arrivée au laboratoire, c'est à cette activité que nous nous sommes d'abord intéressés. La raison de ce choix est qu'un chercheur dans ce domaine nous avait demandé de lui concevoir un programme.

Ce progamme devait réaliser un interfaçage entre un photomètre et un micro-ordinateur de type PC. Un photomètre est un appareil qui permet de mesurer la cinétique de production de lumière par certaines substances biologiques et qui fournit des données sur ces mesures. De telles mesures peuvent notamment servir à connaître la concentration de substances biologiques dans un échantillon.

Lors de l'acquisition de l'appareil, un programme a été fourni. Ce programme permet le dosage de l'Adénosine TriPhosphate (ATP). Pour un type de réaction particulier, le programme commence par réaliser des intégrations sur un certain temps et à étalonner l'appareil. Il réalise ensuite un dosage de l'ATP chaque fois qu'on soumet à l'appareil un échantillon ayant ce type de réaction. Ce programme n'est pas intéressant pour les chercheurs du laboratoire, car il s'agit d'une application trop restreinte et trop spécifique.

En outre, une table traçante est branchée sur le photomètre. Elle permet de dessiner, pendant le test, une courbe représentant l'intensité lumineuse de la substance en fonction du temps. A la fin d'un test, l'utilisateur effectue manuellement, sur cette courbe, des calculs de moyennes, de sommets et d'intégrales en certains points de la courbe. Ces opérations sont assez fastidieuses.

Nous avons donc conçu un programme d'interfaçage plus général et plus proche des besoins des utilisateurs, puisque spécifié par eux. Ce programme est utilisé lors d'une expérience avec le photomètre, une expérience pouvant se composer de plusieurs tests, un test pouvant comprendre plusieurs épreuves. Une épreuve est la mesure par l'appareil de la cinétique de la production lumineuse d'une substance pendant un laps de temps déterminé par l'utilisateur. Les différentes épreuves d'un même test se déroulent sur un même échantillon. Ce programme permet :

- de recevoir des données de l'appareil et de construire pendant une épreuve, en temps réel, un graphique exprimant l'intensité lumineuse en fonction du temps, avec changements d'échelle suivant la longueur de l'épreuve
- à la fin d'une épreuve, lorsque l'appareil s'arrête d'émettre, de calculer la pente, le sommet et l'intégrale de la courbe
- d'effectuer des "zooms" sur des parties de courbes et d'effectuer les mêmes calculs sur ces "zooms"
- de construire des graphes, pour une expérience, dont chaque point est la moyenne pour un test de l'intégrale, du sommet ou de la pente des épreuves constituant le test
- de stocker des données concernant les expériences sur disquette.

Dans le cadre de notre prospection de possibilités de travail sur base des activités du laboratoire, nous avons abordé la bioluminescence, et en première approche, le mémoire réalisé en 1985 par un chercheur du laboratoire [ROG85]. Ce mémoire, intitulé "Mise au point d'un biosenseur pour le dosage du NADH", visait à mettre au point un dispositif permettant de sonder spécifiquement la présence d'une substance organique et de déterminer sa concentration dans un mélange de substances. Un tel dispositif s'appelle un biosenseur. Il se compose de trois parties :

- un système enzymatique immobilisé
- un traducteur qui varie selon le système enzymatique. Ce traducteur peut par exemple être un photomètre
- un analyseur qui transforme les données en informations directes.

Le travail réalisé dans le cadre du mémoire, a consisté à mettre au point et à optimiser le système enzymatique immobilisé.

La bioluminescence, en général, est utilisée au laboratoire pour le dosage de substances.

Nous avons, pour tenter de comprendre ce sujet, eu plusieurs entrevues avec l'auteur du mémoire, nous avons lu le mémoire dans les grandes lignes et nous avons, avec notre directeur de mémoire, assisté à un séminaire de présentation. Finalement nous sommes arrivés à la conclusion qu'il ne serait pas intéressant, ni pour nous, ni pour les biologistes, de développer un outil de type "système expert" sur base de ce sujet. Il est en effet très difficile de pouvoir dégager et modéliser le processus de raisonnement sous-jacent aux décisions relatives aux tâches de dosage de substances. Ces décisions relèvent en grande partie du "feeling" du chercheur.

1.3. LA PURIFICATION DE SUBSTANCES

A ce moment, le directeur du laboratoire nous a parlé d'une tâche qui est exécutée dans tous les domaines de travail du laboratoire : la purification de substances d'origine biologique. Cette tâche est souvent un préalable obligé au travail des chercheurs.

Le biochimiste se trouve en présence d'un mélange de substances d'origine biologique à purifier. Il dispose de techniques de purification qu'il doit combiner afin d'obtenir la séquence qui lui donnera la meilleure purification possible.

Il existe, à l'heure actuelle, très peu d'écrits rassemblant des connaissances sur la manière de combiner ces techniques de purification. Les seuls écrits existant présentent le plus souvent l'application d'une combinaison de ces techniques pour un mélange. Cette application permet d'optimiser la purification du mélange et est le fruit de recherches dans le domaine des purifications. Les biochimistes appliquent par la suite cette combinaison comme une "recette", avec certaines adaptations mais sans remettre en cause le principe. Lorsqu'il n'existe aucun écrit, les biochimistes conçoivent eux-mêmes des combinaisons des techniques de purification pour les mélanges qu'ils ont à purifier. Ils procèdent le plus souvent "par essais et erreurs" et appliquent des connaissances acquises par expérience. On ne peut pas dire qu'ils disposent vraiment de critères pour le choix systématique des techniques et pour leur combinaison.

Il nous a semblé qu'un outil de type système expert pourrait aider le biochimiste dans cette tâche de conception de plans d'expérience.

A partir de ce moment, nous avons commencé une étroite collaboration avec les experts en purification du laboratoire, et nous avons joué le rôle d'ingénieurs de connaissances. Cette collaboration a consisté en de nombreuses entrevues, en la projection de films sur le sujet, en assistance à des cours à la Faculté de Biologie. A l'heure actuelle, on peut dire que, malgré le fait que nous ayons circonscrit le problème à trois techniques de purification parmi plusieurs disponibles, cette phase d'acquisition des connaissances n'est pas terminée.

Nous avons appris plus tard que l'idée d'une approche informatique de cette tâche avait également germé ailleurs. La firme Pharmacia, installée à Uppsala en Suède, a en effet

conçu un programme de simulation de l'expérience de purification de substances. Ce programme est destiné à fournir une aide pour l'apprentissage des techniques de purification. Il présente à l'utilisateur un mélange fictif contenant vingt protéines. L'utilisateur peut choisir la protéine qu'il désire purifier. Il décide ensuite des techniques de purifications qu'il va appliquer, parmi celles qui lui sont disponibles. Après chaque choix d'une technique, le programme fournit à l'utilisateur une description du mélange obtenu et de l'efficacité de sa purification, notamment.

Des chercheurs de cette même firme ont aussi entrepris une approche "système expert" de ce problème.

CHAPITRE 2. LA PURIFICATION DE SUBSTANCES D'ORIGINE BIOLOGIQUE

Lorsque nous nous sommes engagés dans l'étude d'un système d'aide à la purification de substances d'origine biologique, notre rôle s'est limité, dans un premier temps, à nous familiariser avec le domaine et les problèmes qu'on y rencontre. Nous présentons dans ce chapitre notre première approche du domaine. Il s'agit plutôt d'une approche théorique dans laquelle nous avons essayé d'étudier les méthodes biochimiques de purification et les mélanges auquels ces méthodes sont appliquées, de comprendre comment se déroule une purification dans un laboratoire.

Ce chapitre est important pour la compréhension du problème par un informaticien. Cette approche du problème est nécessaire pour rendre possible la conception d'un outil informatique. Cependant, elle peut paraître inutile ou incomplète, à une personne déjà familiarisée avec le domaine.

2.1. DESCRIPTION DU DOMAINE

L'histoire de la biochimie est dans une large mesure liée à l'histoire de l'isolation de substances biologiques à partir de tissus cellulaires. Une partie importante du travail en biochimie nécessite l'isolation et la séparation de composants d'origine biologique. Le succès des recherches dépend d'une préparation judicieuse des substances cellulaires.

Dans la plupart des cas, le matériel de départ est complexe. Un mélange biochimique est composé d'un grand nombre de substances, notamment des protéines, ayant des

propriétés chimiques assez similaires. Ces substances sont sensibles à de nombreux facteurs chimiques et physiques. Des conditions physiques inadéquates, telles que des températures inappropriées par exemple, peuvent changer la structure des substances de manière irréversible et détruire leur activité biologique.

La difficulté du travail de purification réside dans la mise au point et l'utilisation de techniques raffinées de séparation. Dans son laboratoire, le biochimiste doit réaliser un choix judicieux dans les méthodes de séparation.

Les techniques de purification sont utiles pour les travaux analytiques et préparatifs, au niveau de la recherche et de l'industrie. Les travaux analytiques ont pour objectif l'identification et la caractérisation des diverses substances d'un mélange. Dans les travaux préparatifs, les techniques servent à purifier des substances afin de les utiliser par la suite dans d'autres travaux.

2.2. TECHNIQUES DE PURIFICATION

De nombreux développements techniques ont été nécessaires pour mettre au point des méthodes de séparation efficaces. Les méthodes utilisées aujourd'hui sont principalement l'électrophorèse, la précipitation isoélectrique, la chromatographie sur échangeur d'ions, la chromatographie sur tamis moléculaire, la chromatographie d'affinité, la chromatographie en phase gazeuse, etc. Elles séparent les substances sur base de propriétés physiques différentes des molécules telles que la masse, la taille, la charge électrique, la solubilité, etc. Par exemple, une technique de purification basant sa séparation sur la masse

moléculaire des substances sépare d'autant mieux les substances que leurs masses moléculaires sont différentes. La purification obtenue par l'utilisation d'une technique n'est en général que partielle. Les substances ayant une même valeur de la propriété physique utilisée par la technique ne peuvent être séparées. La purification optimale d'une substance d'un mélange complexe requiert donc l'application adéquate d'une suite de plusieurs de ces techniques.

Des purifications ont été mises au point. Elles utilisent les techniques de purification pour isoler une substance particulière dans un mélange complexe. Ces purifications sont présentées dans la littérature. Ce sont des "recettes" où l'on expose, étape par étape, les techniques de purification à appliquer et les résultats obtenus. Ces purifications présentent rarement les raisonnements qui ont accompagné leur mise au point. Cette lacune peut parfois s'expliquer par l'impossibilité dans laquelle se trouve le chercheur de justifier clairement l'efficacité de sa purification. Un chercheur peut très bien constater "par essais et erreurs" l'efficacité d'une purification qu'il élabore sans pouvoir l'expliquer dans l'immédiat. Les composants d'un mélange dont on souhaite extraire une substance sont parfois mal connus. Leurs réactions peuvent donc échapper au chercheur. L'explication peut apparaître plus tard après beaucoup de recherches.

Lorsqu'un biochimiste doit réaliser la purification d'une protéine, il recourt aux connaissances accumulées dans son laboratoire, ou à la littérature. Il connaît en théorie les techniques de purification disponibles dans son laboratoire et sait les utiliser avec plus ou moins de compétence. Il connaît les composants du mélange avec plus ou moins de précision. En appliquant une purification présentée dans la littérature, un biochimiste peut avoir

des difficultés pour étudier et évaluer les raisonnements qui ont guidé l'élaboration de cette purification. Il peut l'appliquer strictement comme une "recette". Cette restriction ne constitue pas un problème majeur lorsque la purification présentée est immédiatement applicable et efficace. Malheureusement, comme une "recette", la purification présentée requiert un savoir-faire, une pratique des techniques de purification qui ne sont pas exposés dans la littérature. Aussi, la purification présentée n'est souvent qu'une base. En essayant de l'appliquer étape par étape, le biochimiste constate très souvent une efficacité moindre que celle qu'il souhaite. Il doit alors ajuster lui-même la purification avec précision aux conditions de travail qui sont les siennes. Il doit utiliser le matériel disponible dans son laboratoire. Le travail qu'il souhaite réaliser sur la substance requiert une purification particulière. L'adaptation d'une purification à des conditions pratiques de travail n'est pas simple, demande un certain temps et un savoir-faire.

Les purifications présentées dans la littérature ne sont pas une source suffisante de connaissances pour un biochimiste. Très souvent, le chercheur fera appel à l'aide de personnes expérimentées dans les diverses techniques de purification. Ces personnes peuvent certainement apporter une expérience pratique indispensable. Cependant, ces personnes ne sont pas toujours disponibles. Chaque laboratoire ne dispose pas de spécialistes en purifications. De plus, les connaissances sont distribuées chez un grand nombre de personnes, car chacune a une expérience plus ou moins importante dans l'utilisation de certaines techniques et dans la manipulation de certains mélanges.

2.3. PRINCIPE DE LA CHROMATOGRAPHIE

Parmi les techniques de purification, les plus efficaces sont des chromatographies. Presque sans exception, une purification inclut l'utilisation d'une chromatographie.

Découverte au début du siècle par l'observation de la séparation des composants de la chlorophylle en zones colorées, la chromatographie s'est généralisée par la suite à la séparation de substances incolores, allant des gaz aux composés macromoléculaires, organiques ou minérales. Dès lors, son nom ne se justifie guère.

Lors d'une chromatographie, le système est composé de trois éléments : l'échantillon à purifier, la phase stationnaire et la phase mobile. L'échantillon, entraîné avec la phase mobile par gravité, pression ou d'autres moyens mécaniques, migre au travers de la phase stationnaire. Celle-ci a pour mission de retenir certaines substances du mélange ou de ralentir leur progression. Les différentes substances du mélange ont une affinité différente pour la phase stationnaire en fonction de la propriété physique des substances sur laquelle se base la chromatographie. Les différentes substances du mélange sont retenues ou retardées différemment. Elles ne migrent pas de la même manière et donc se séparent. Deux substances sont séparées pour autant qu'elles aient des propriétés différentes vis-à-vis des deux phases en présence. L'affinité pour les deux phases peut être estimée par un coefficient de partage.

La chromatographie, technique analytique à l'origine, est devenue un procédé de préparation très pratique pour des substances diverses. Elle a également été adaptée aux préparations industrielles.

2.4. DESCRIPTION DE TECHNIQUES DE PURIFICATION

Dans notre première approche du domaine, nous avons décidé de nous intéresser plus particulièrement à trois techniques de purification : la chromatographie sur tamis moléculaire, la chromatographie sur échangeur d'ions et la précipitation isoélectrique. Ces techniques ont été choisies pour leur complémentarité. En outre, elles sont souvent utilisées au laboratoire de biochimie. Nous nous sommes également intéressés à des techniques biochimiques utilisées pour compléter la réalisation des purifications : la dialyse et la concentration sur membrane d'ultra-filtration.

2.4.1. Définitions

2.4.1.1. Le pH

Le pH d'une solution est une mesure de la concentration en ions H_3O^+ de cette solution :

$$pH = - \log [H_3O^+].$$

En fait, le pH permet de caractériser l'acidité ou la basicité d'une solution. A 25°C par exemple, la concentration en H_3O^+ d'une solution neutre est égale à $10^{-7}M$ et le pH est égal à 7. Dans une solution acide, la concentration en H_3O^+ est supérieure à la concentration en OH^- et le pH est inférieur à 7. Dans une solution basique, la concentration en H_3O^+ est inférieure à la concentration en OH^- et le pH est supérieur à 7.

2.4.1.2. Le tampon

Lorsqu'on manipule des substances biochimiques comme c'est le cas lors d'une purification, il est important de stabiliser le pH du système. Les substances biochimiques, comme les enzymes par exemple, sont sensibles aux changements de pH. Des variations de pH peuvent dénaturer les substances. On doit dès lors disposer de milieux capables de garder un pH déterminé et constant malgré l'addition d'ions H_3O^+ ou OH^- dans le système. Ces milieux sont appelés les tampons.

Un tampon dispose d'une capacité plus ou moins grande à résister à une variation importante du pH. Cette capacité est applée le "pouvoir tampon". A chaque tampon est associée une zone de "pouvoir tampon". Il s'agit d'une zone de pH à l'intérieur de laquelle le tampon contrôle bien les fluctuations de pH. Elle est définie par l'intervalle [pK-1,pK+1]. Le pK d'un tampon est le pH auquel une forme acide (ou basique) est à moitié dissociée.

2.4.2. LA CHROMATOGRAPHIE SUR TAMIS MOLECULAIRE

2.4.2.1. Principe

La chromatographie sur tamis moléculaire est une méthode de séparation de substances biologiques basée sur la différence de tailles moléculaires.

Pour obtenir une séparation, l'opérateur utilise un gel. Celui-ci est livré sous forme de poudre, puis gonflé dans l'eau pour son utilisation. L'opérateur obtient ainsi une structure tri-dimensionnelle formée de grains possédant des pores de dimensions variables. Le gel constitue la phase stationnaire de la chromatographie. Il est placé dans

un tube qui constitue une colonne chromatographique. Il est ensuite équilibré à l'aide d'un tampon (liquide qui servira à l'élution des protéines).

Sur la colonne chromatographique, l'opérateur verse la substance biologique à purifier. Il raccorde ensuite à la colonne un récipient contenant un liquide (tampon) qui va traverser le gel en entraînant la substance. Il s'agit du phénomène d'élution. Ce liquide est appelé l'éluant et il constitue la phase mobile de la chromatographie. Le débit du liquide est contrôlé par une pompe péristaltique à la sortie de la colonne.

Les composants du mélange dont les molécules sont de dimension supérieure aux plus gros pores du gel ne peuvent pénétrer dans les grains du gel. Ils traversent la colonne avec l'éluant et sortent les premiers. Par contre, les composants dont les molécules sont plus petites sont freinés par la phase stationnaire. Ils diffusent plus ou moins dans le gel suivant la taille et la forme de leurs molécules, et ils traversent plus lentement la colonne. Les composants du mélange quittent donc la colonne dans l'ordre des masses moléculaires décroissantes. Ce phénomène est illustré par la figure 2.1. [PHAa].

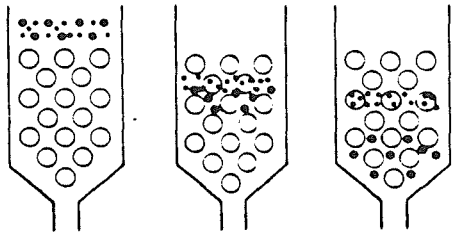

Figure 2.1.

A la sortie de la colonne, une mesure est effectuée sur les quantités éluées, par enregistrement, de façon continue, de l'absorption de lumière des protéines à 280nm. On peut ainsi évaluer la concentration des différents protéines qui sortent de la colonne. Les quantités éluées sont recueillies par un collecteur de fractions qui les répartit dans différents tubes.

A la fin de l'expérience, la colonne est lavée et est prête pour un usage ultérieur.

2.4.2.2. Concepts théoriques

A partir des résultats enregistrés, la variable la plus importante à déterminer est Ve, le volume d'élution d'une substance. C'est le volume de la phase liquide nécessaire à l'élution de la substance.

Un autre concept important est V0, le volume mort ou volume vide. C'est le volume du liquide qui peut circuler entre les grains du gel. Il est donné par le volume d'élution d'une substance exclue du gel. V0 est une constante pour une colonne et un gel donnés.

Vt est le volume total de la colonne ou le volume de la phase liquide à l'intérieur et à l'extérieur du gel. Il est donné par le volume d'élution d'une substance qui peut diffuser dans tout le gel. Vt est aussi une constante pour une colonne et un gel donnés.

Ces données, seules, n'ont qu'une signification limitée. On les utilise en fait pour calculer le coefficient de partage

$$Kav = Ve - V0 / Vt - V0.$$

Kav exprime l'affinité d'une substance pour la phase stationnaire. Le coefficent de partage pour une substance (compris entre 0 et 1) est d'autant plus petit que cette substance est grosse en terme de taille moléculaire, et donc exclue du gel. Plus la taille des molécules d'une substance est importante, plus son volume d'élution est proche du volume mort.

Dans le domaine des protéines, de nombreux travaux ont montré que le comportement chromatographique dépend du poids moléculaire des substances. Il existe des représentations graphiques de cette relation appelées courbes d'étalonnage. Ces courbes, établies par chromatographie de protéines connues, expriment le coefficient de partage comme une fonction sensiblement linéaire du logarithme de la masse moléculaire. Elles permettent aussi une bonne représentation des possibilités de travail avec les différents gels.

La figure 2.2. donne les courbes d'étalonnage typiques de trois types de gels [FIS80].

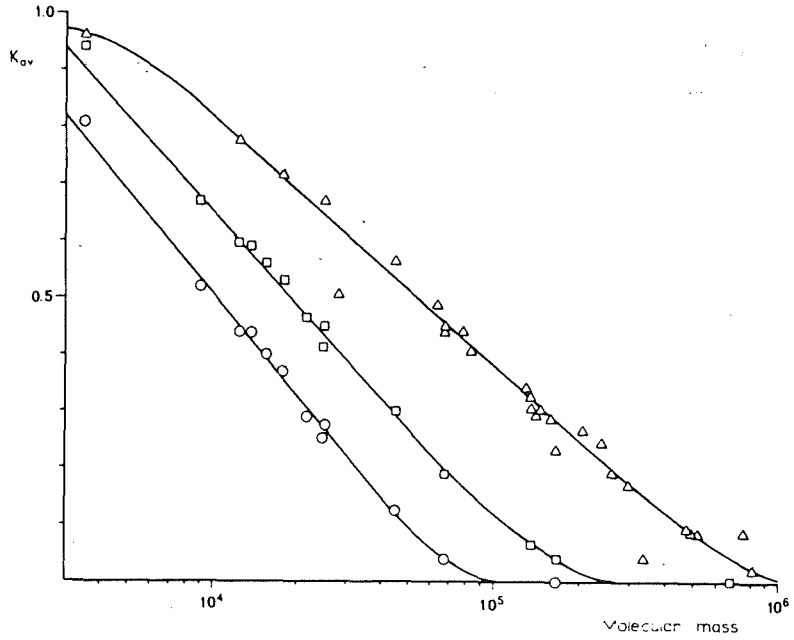

Figure 2.2.

En-dessous d'une certaine masse moléculaire, la courbe devient horizontale. Dans cette région, toutes les substances sont éluées ensembles avec un volume d'élution proche du volume total de la colonne (Kav proche de 1).

Dans la partie de courbe inclinée, une variation de la masse moléculaire correspond à une variation du coefficient de partage, donc du volume d'élution, entraînant une séparation des substances de masses moléculaires différentes. Cette partie de la courbe représente le domaine de fractionnement du gel, c'est-à-dire la zone de poids moléculaires où le gel va réaliser une séparation.

Pour les masses moléculaires au-dessus du domaine de fractionnement, la courbe est à nouveau horizontale. Dans

cette zone, toutes les substances sortent avec le volume mort, ce qui correspond à un coefficient de partage égal à 0. Les molécules sont trop grosses et elles ne peuvent pénétrer dans le gel. Le point où Kav atteint 0 (Ve égal à V0) représente la limite d'exclusion du gel.

2.4.2.3. Application : séparation des différents constituants d'un produit

En chromatographie, l'analyse quantitative peut être effectuée en enregistrant de façon continue l'évolution d'une caractéristique physique des substances éluées (absorption de lumière, activité optique, indice de réfraction, etc.). Les substances éluées peuvent aussi être récupérées par fractions et le dosage effectué sur chacune d'elles suivant des critères physiques, chimiques ou biologiques propres à chaque substance.

Au laboratoire, comme nous l'avons déjà expliqué, l'opérateur réalise une mesure continue de l'absorption de lumière pour les protéines. On obtient ainsi une courbe de mesure de la concentration de protéines en fonction du volume de liquide élué, telle que celle de la figure 2.3. Pour chaque protéine du mélange, la courbe dessinée croît à partir d'une concentration nulle, atteint un sommet, et décroît jusqu'à une concentration nulle. En abscisse du "pic" d'une protéine, on peut déterminer le volume d'élution de cette protéine [FIS80].

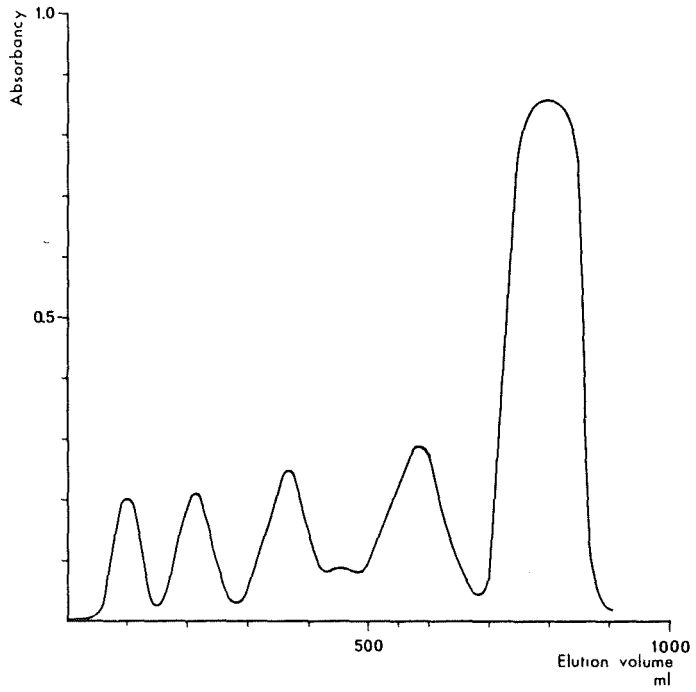

Figure 2.3.

Supposons que, dans un mélange de protéines, on souhaite obtenir une protéine dont nous connaissons le poids moléculaire.

Avant de réaliser la séparation du mélange, il faut établir une courbe d'étalonnage de la colonne. En effet, les courbes d'étalonnage théoriques ne peuvent être généralisées à toutes les colonnes. Il faut établir une courbe d'étalonnage propre à la colonne utilisée. Pour ce faire, l'opérateur passe dans la colonne, et séparément, différentes protéines pures de poids moléculaire connu. Pour chaque protéine se dessine une courbe de la concentration en fonction de l'élution. En abscisse du "pic" de la protéine, on découvre le volume d'élution Ve. Ayant aussi déterminé le volume mort et le volume total en utilisant une substance de haut poids moléculaire et une substance de faible poids moléculaire, on peut calculer le coefficient de partage Kav de la protéine. Le couple (poids moléculaire, coefficient de partage) est un point de la courbe d'étalonnage.

L'opérateur effectue ensuite la séparation proprement dite du mélange. Il obtient une courbe exprimant la concentration en fonction de l'élution. Si la séparation s'est bien faite, cette courbe compte un "pic" par composant séparé et chaque composant séparé est récolté dans une ou plusieurs éprouvette(s). Puisque le poids moléculaire de la protéine à isoler est connu, l'opérateur peut, à partir de la courbe d'étalonnage, trouver son coefficient de partage Kav et donc son volume d'élution Ve. Il utilise alors le volume d'élution Ve pour déterminer le "pic" et l'éprouvette correspondant à la protéine à isoler.

Afin que la séparation se réalise convenablement, il faut que les protéines constituant le mélange aient des volumes d'élution les plus différents possibles.

2.4.2.4. Application : détermination des masses moléculaires

La filtration sur tamis moléculaire permet, sans purification poussée, d'estimer le poids moléculaire de protéines.

Supposons qu'on souhaite connaître le poids moléculaire d'une protéine contenue dans un mélange.

On établit une courbe d'étalonnage de la colonne. Ne connaissant pas le poids moléculaire de la protéine, on ne peut pas déterminer son volume d'élution. Pour le connaître, il suffit de mesurer l'activité de la protéine dans les différentes éprouvettes contenant les substances éluées. On peut ainsi découvrir quelle(s) éprouvette(s) contient(nent) la protéine et déterminer son volume d'élution sur la courbe de concentration en fonction de l'élution.

Grâce au volume d'élution, on peut calculer le coefficient de partage Kav de la protéine et consulter la courbe d'étalonnage pour connaître son poids moléculaire.

2.4.2.5. Application : déssalage

Le déssalage consiste à éliminer les sels, substances de faible poids moléculaire, d'un mélange.

Pour éliminer les sels et les substances de faible poids moléculaire d'un mélange, la technique du tamis moléculaire s'avère très efficace.

Pour cette opération, un gel particulier est utilisé : le Sephadex G25. Son domaine de fractionnement ne couvre que de faibles poids moléculaires, inférieurs à 5000 dalton. Les substances de poids moléculaire élevé, qui doivent être déssalées, migrent dans le volume mort. Les composants de faible poids moléculaire, comme les sels, se répartissent entre la phase stationnaire et la phase mobile. La séparation entre composants lourds et composants légers est très bonne.

2.4.2.6. Condition d'application : volume du mélange

Lorsqu'on dépose un mélange de protéines sur une colonne, chaque protéine migre en s'étalant dans le gel. Ce phénomène d'étalement des protéines est d'autant plus important que le volume du mélange est grand. Il est néfaste à la séparation des protéines car un étalement trop important implique un recouvrement des "pics" des différentes protéines. Aussi l'opérateur ne peut déposer un volume trop important sur une colonne.

2.4.2.7. Condition d'application : concentration du matériel dans le mélange

La viscosité du mélange limite la concentration. Une viscosité élevée provoque un débit irrégulier, un étalement non symétrique des protéines. Pour éviter ces problèmes, une concentration de 10mg de matériel par ml est conseillée. C'est une concentration "idéale". On peut cependant utiliser un tamis moléculaire avec des concentrations plus faibles, par exemple comprises entre 2 et 10mg par ml. Des concentrations moins importantes augmentent les pertes relatives qui se produisent sur les parois et dans le gel.

2.4.3. LA CHROMATOGRAPHIE SUR ECHANGEUR D'IONS

2.4.3.1. Principe

La chromatographie sur échangeur d'ions est une des méthodes les plus importantes pour la séparation des substances biologiques. Le principe de base de l'échange d'ions est de réaliser une séparation sur base des différentes charges électriques des molécules.

Un échangeur d'ions est une matrice insoluble (gel, résine) sur laquelle on a fixé des groupements chimiques chargés. S'il s'agit de groupements négatifs, nous avons un échangeur de cations (ions positifs), c'est-à-dire qu'un échangeur chargé négativement fixera les molécules chargées positivement. S'il s'agit de groupements positifs, nous avons un échangeur d'anions (ions négatifs), c'est-à-dire qu'une matrice chargée positivement fixera les molécules chargées négativement.

Une expérience comprend généralement quatre étapes : l'équilibrage de l'échangeur d'ions, l'addition et la fixation des substances de l'échantillon, le changement des conditions pour obtenir une désorption sélective, la régénération de l'échangeur d'ions.

L'équilibrage de l'échangeur d'ions se fait à l'aide du tampon (liquide qui servira durant l'expérience à l'élution des protéines) pour obtenir les meilleures conditions de pH et de force ionique pour l'expérience.

Une fois l'équilibrage réalisé, on place l'échantillon sur la colonne. Les substances de charge opposée à la charge de la colonne vont se fixer sur l'échangeur par adsorption. Les autres substances vont être éluées de la colonne par un tampon.

La troisième phase d'une expérience est la désorption des substances fixées sur la colonne. Cette désorption va être effectuée par une modification du pH qui va entraîner une variation des charges des différentes protéines fixées ou par une augmentation de la force ionique qui aura pour conséquence une compétition entre les ions et les protéines fixées. Ces modifications des conditions peut s'effectuer de manière continue (on parle alors d'un gradient) ou de manière discrète. Une séparation est ainsi obtenue puisque les différentes protéines ont des affinités différentes pour l'échangeur, du fait de leurs charges respectives.

La dernière étape consiste en une régénération de l'échangeur pour une utilisation ultérieure.

2.4.3.2. Concepts théoriques

Il existe une relation entre la charge d'une protéine et son pH. Cette relation est schématisée par la figure 2.4. [PHAb].

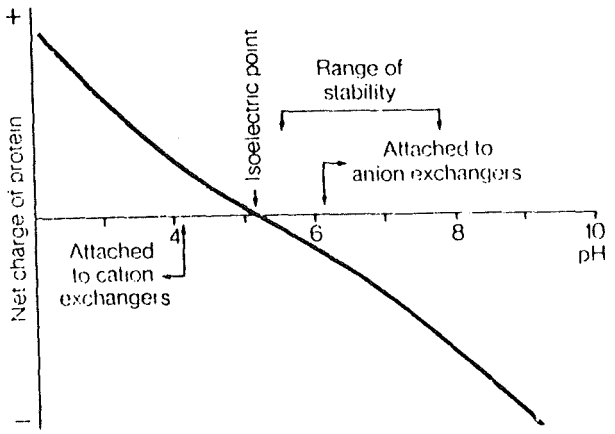

Figure 2.4.

Le point où la fonction croise l'ordonnée est appelé point isoélectrique. Ce point exprime le pH pour lequel la protéine est globalement neutre. Pour les valeurs de pH inférieures à ce point, la protéine est chargée positivement, alors qu'elle est chargée négativement pour les valeurs de pH supérieures.

Le domaine de stabilité d'une protéine est l'intervalle de valeurs de pH dans lequel cette protéine est stable. Les expériences doivent se dérouler à un pH compris dans ce domaine. La stabilité est une notion très relative, pour laquelle il n'existe pas de critère universel. Les protéines s'altèrent au cours du temps. Une substance est stable lorsqu'elle conserve beaucoup de son activité immunologique ou enzymatique.

2.4.4. LA PRECIPITATION ISOELECTRIQUE

2.4.4.1. Principe

Cette technique est basée sur le fait qu'une substance précipite lorsque la concentration d'un agent de précipitation atteint une certaine valeur. Les molécules de cette substance précipitées peuvent être récupérées après centrifugation sous forme d'un culot.

Au fur et à mesure que la concentration de l'agent augmente, les substances contenues dans le mélange précipitent les unes après les autres, suivant leur solubilité.

2.4.4.2. Concepts théoriques

En augmentant la concentration en sels dans un milieu, l'opérateur augmente la force ionique μ du milieu :

$$\mu = 1/2 \; \Sigma \; Ci \; Zi^2$$

où

Ci = concentration d'un ion i

Zi = charge d'un ion i.

La solubilité d'une substance s'exprime en mg/ml. Elle est plus faible aux très faibles et aux fortes forces ioniques. Aux forces ioniques moyennes, le logarithme de la solubilité est proportionnel à la force ionique :

$$\log S = B - K\mu$$

où

K, la pente de la droite est très semblable pour un même agent de précipitation

B, la constante de solubilité varie en fonction de la nature de la substance.

La figure 2.5. illustre la solubilité de diverses protéines dans des solutions où l'agent de précipitation est le sulfate ammonique [REM86].

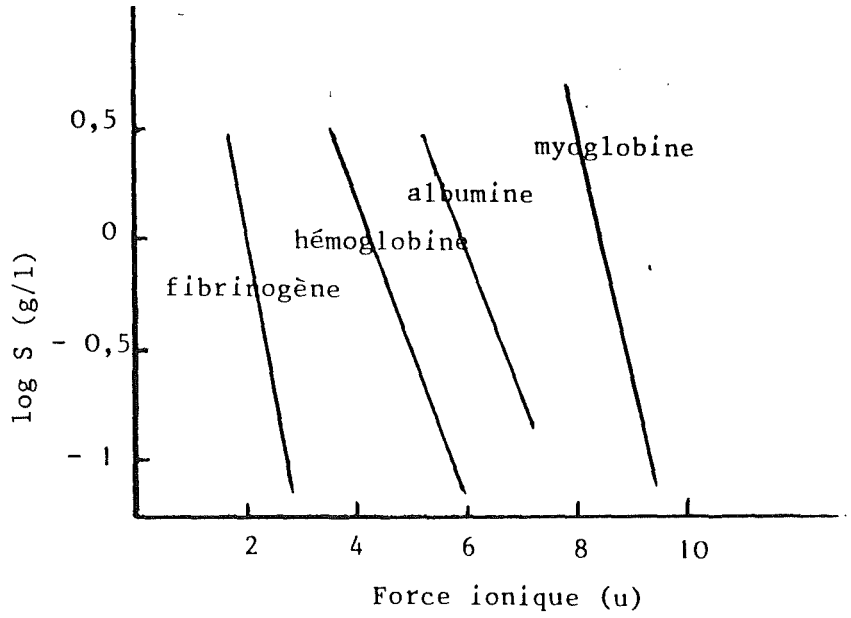

Figure 2.5

2.4.4.3. Application de la technique

Le mélange de protéines à purifier est dilué et versé dans un récipient. L'opérateur ajoute progressivement un agent de précipitation. On utilise souvent une solution de sulfate ammonique saturée à 4°C et à pH 7.4. La force ionique du milieu augmente donc progressivement. Les différentes protéines du mélange précipitent à différentes valeurs de force ionique..

Si l'opérateur connaît le coefficient K correspondant au sulfate ammonique et la constante de solubilité B pour les différentes protéines du mélange, il pourra estimer la force ionique nécessaire à précipiter la protéine qu'il souhaite purifier.

Par exemple, supposons un mélange des quatre protéines de la figure ci-dessus. Si l'opérateur veut purifier l'hémoglobine, il sait qu'il doit augmenter la force ionique jusqu'à 6 afin que cette protéine soit entièrement précipitée. Cependant, il n'obtiendra pas l'hémoglobine pure, mais un mélange dans lequel se trouverons deux types de contaminants : d'abord la contamination des protéines déjà précipitées, le fibrinogène et l'albumine en partie, ensuite la contamination due à la présence des protéines non encore précipitées présentes dans tout le volume du liquide et donc également dans le culot. L'opérateur peut considérer que la force ionique est suffisante lorsque le log S de la protéine à purifier devient inférieur à -1, ce qui correspond à moins de 0.1 g/l.

La technique est utilisée soit pour précipiter la protéine à purifier et récupérer le culot, soit pour précipiter un certaine nombre de contaminants et récupérer le surnageant.

Remarquons cependant que le coefficient K de l'agent de précipitation et la constante de solubilité B des différentes protéines pour un agent de précipitation sont peu connus et peu disponibles dans la littérature. Dans un laboratoire, un opérateur ne connaît généralement que les zones approximatives de précipitation de quelques protéines.

Cette technique est simple à utiliser. Elle ne permet pas de purifier parfaitement une protéine, mais elle concentre les protéines. Elle constitue une méthode complémentaire aux chromatographies. La précipitation est particulièrement intéressante lorsque la quantité de matériel est importante (du gramme au kilo). Une première purification peut dans ce cas être réalisée rapidement en utilisant un minimum de moyens (récipient , agent de

précipitation). Avec des quantités importantes de matériel, les autres techniques ne sont pas toujours adaptées, ou encore impliquent des coûts importants.

2.4.4.4. Volume de l'agent de précipitation

L'agent de précipitation est un sel. Une trop forte présence en sels risque de dénaturer les protéines. De plus, la présence de sels est exclue lorsqu'on envisage certaines techniques de purification, comme la chromatographie sur échangeur d'ions par exemple. Les sels doivent donc souvent être éliminés après une précipitation isoélectrique. Une trop forte concentration en sels pose des problèmes pour son élimination. Aussi, l'opérateur doit-il limiter le volume de solution en sels qu'il ajoute au mélange lorsqu'il réalise la précipitation. La solution finale, obtenue après la précipitation, doit être composée au maximum de 60% de la solution en sels et de 40% du mélange de départ.

2.4.5. La dialyse

La dialyse est une technique de séparation utilisée pour purifier des substances d'origine biologique telles que les protéines, les enzymes, les hormones.

2.4.5.1. Principe

Le phénomène fondamental qui est à la base de la dialyse est la diffusion. Une solution aqueuse d'un mélange, sur laquelle on dépose de l'eau dans une éprouvette, permet de comprendre le phénomène de diffusion : certaines substances de la solution diffusent rapidement

dans l'eau, qui joue le rôle de solvant. Le coefficient de diffusion d'une substance en solution dépend essentiellement de ses dimensions : plus la molécule est grosse, plus faible est le coefficient de diffusion. Une séparation se réalise ainsi entre les substances qui diffusent rapidement et les substances qui diffusent lentement.

La séparation est encore plus nette si on interpose une membrane entre la solution et le solvant. Les membranes généralement utilisées sont poreuses. Les grosses molécules, qui ont dès le départ un faible coefficient de diffusion, sont arrêtées par la membrane, alors que les petites molécules peuvent la traverser facilement. La dialyse est une technique de séparation par diffusion à travers une membrane.

D'autres facteurs que la taille des molécules peuvent intervenir dans la séparation par dialyse. Certaines membranes chargées électriquement peuvent repousser des ions de même signe. Il se peut aussi que la membrane ne soit pas poreuse. La diffusion ne s'effectue que si la substance peut se dissoudre dans la membrane. La température peut exercer une influence : en présence de fortes températures, le phénomène de diffusion est accentué. Cependant, si les substances sont des protéines, la dialyse doit se dérouler dans une pièce froide pour minimiser la dénaturation.

2.4.5.2. Application de la technique : le déssalage

Cette application est présentée telle qu'elle est réalisée au laboratoire.

Lorsqu'un mélange contient de grosses molécules, telles que les molécules de substances d'origine biologique (les protéines, par exemple), et de petites molécules, telles que

celles des sels par exemple, la dialyse peut être appliquée pour débarasser le mélange des sels qu'il contient.

La solution contenant le mélange à déssaler est placée dans un sac à dialyse. Une membrane poreuse constitue les parois de ce sac. L'ensemble est plongé dans un volume d'eau qui joue le rôle de solvant. Le sel contenu dans le mélange va diffuser dans le solvant.

Le déssalage par dialyse est une technique annexe aux techniques chromatographiques et à la précipitation qui sont fréquemment utilisées. En effet, l'application d'une technique de purification peut entraîner une augmentation de la concentration en sels qui doit être réduite avant l'application de la technique suivante.

La dialyse est un procédé simple et peu onéreux. Elle est cependant généralement lente. Pour en raccourcir la durée, le choix des conditions expérimentales est important. L'épaisseur de la membrane doit être faible. Le volume de solvant doit être grand, égal au moins à 50 fois le volume de la solution à dialyser. L'efficacité est encore accrue en renouvellant constamment le solvant.

Dans le même but d'accélérer la dialyse, on peut, dans certains cas, associer l'action d'un champs électrique au phénomène de diffusion. Le champs électrique agit de manière à attirer les petites substances à travers la membrane. On utilise pour ce faire un électrodialyseur.

2.4.6. La concentration sur membrane d'ultra-filtration

La concentration sur membrane d'ultra-filtration est une technique utilisée pour concentrer les mélanges

d'origine biologique et éliminer les substances de faible poids moléculaire.

2.4.6.1. Application de la technique

La solution contenant le mélange est déposée sur une membrane d'ultra-filtration. Celle-ci joue le rôle d'un filtre et laisse passer le solvant et les petites molécules dont le poids moléculaire est inférieur à une certaine limite. L'opérateur peut laisser filtrer le mélange sur la membrane jusqu'au moment où il obtient le volume souhaité. Cette technique de concentration est souvent utilisée lorsqu'on dispose d'un grand volume d'échantillon et qu'on souhaite appliquer une technique de purification qui n'accepte que de faibles volumes, comme la chromatographie sur tamis moléculaire par exemple.

CHAPITRE 3. UN SYSTEME EXPERT POUR LA PURIFICATION DE SUBSTANCES

Nous avons envisagé d'étudier la conception d'un outil de type système expert pour aider à la purification de substances d'origine biologique. Cette étude exigeait l'approfondissement de nos connaissances dans ce domaine. Pour les biochimistes avec lesquels nous avons collaboré, cette approche de leur domaine exigeait une compréhension de base de ce type particulier de programmes. Aussi, nous avons essayé de leur présenter brièvement le domaine de l'intelligence artificielle et plus particulièrement des systèmes experts.

Ce chapitre décrit, dans une première section, ce qu'est un système expert. Cette présentation, très générale, est réalisée à l'intention des personnes non spécialisées dans le domaine. Dans une seconde section, nous envisageons la contribution possible d'un système expert pour une aide à la purification de substances d'origine biologique. Nous présentons les éléments qui nous ont paru justifier l'approche "système expert" pour apporter la contribution d'un outil informatique au domaine.

3.1. UNE APPROCHE DES SYSTEMES EXPERTS

3.1.1. Bref historique de l'intelligence artificielle

Depuis quarante ans, les ordinateurs ont été souvent utilisés pour supporter des logiciels qui étaient des algorithmes, c'est-à-dire des procédures qui, disposant d'informations complètes, garantissent de trouver la

solution correcte à un problème dans un temps déterminé ou qui, du moins, signalent l'absence de solution.

Cependant, à partir des années 50, certains chercheurs ont voulu comprendre le comportement des humains en concevant des programmes dont la conduite se rapproche autant que possible de celle des hommes. On a alors commencé à parler d'intelligence artificielle. Les chercheurs ont pensé, à l'époque, pouvoir développer des machines qui, dans un avenir prévisible, pourraient traiter une étendue de problèmes analogue à celle accessible à l'esprit humain. Leurs ambitions ont été prises au sérieux, à la fois par des organismes gouvernementaux, aux Etats-Unis notamment, et par une partie importante de la communauté scientifique. Ainsi, la psychologie s'est jointe à l'intelligence artificielle pour l'étude de domaines tels que les processus cognitifs sous-jacents à l'acquisition et à la mémorisation de connaissances par l'homme.

Le système de traitement de l'information le plus ambitieux construit pour étudier le comportement humain de résolution de problèmes est le "General Problem Solver" (GPS), de Newell et Simon. De tels progammes étaient appelés "systèmes de résolution de poblèmes". Les prétentions des auteurs du GPS étaient de développer une théorie pouvant expliquer l'homme dans son entier. Ils voulaient ainsi montrer que les ordinateurs pourraient bientôt résoudre des problèmes dans tous les domaines de la pensée humaine.

De nombreux échecs ont contraint les chercheurs à un peu plus d'humilité. Ils se sont rendu compte, vers la fin des années 60, qu'un tel projet n'était pas réalisable. Ils ont compris l'importance d'une connaissance spécifique à un domaine comme base de la résolution de problèmes dans ce domaine. La plupart des problèmes résolus par l'homme ne

pouvaient donc se réduire à des stratégies générales de résolution de problèmes. Par exemple, le problème de la compréhension du langage naturel n'admettait pas de solution générale : le langage n'est compris que dans des cadres contextuels. Pour résoudre des problèmes dans le domaine de l'expertise humaine (médecine, programmation, etc.), la machine devait connaître ce que l'homme qui traite ces problèmes connaît sur le sujet. [HAY84]

A partir de ce moment, progressivement, les programmes de résolution de problèmes se sont limités à l'exécution de tâches spécifiques à un domaine particulier. Ces programmes sont appelés "systèmes experts". On a alors commencé à développer plusieurs applications (Dendral, Mycin, etc.) qui, vers la fin des années 70, ont permis des progrès considérables dans le domaine.

L'intelligence artificielle et les systèmes experts, en particulier, ont atteint, dans les années 80, un point de maturité suffisant pour donner lieu à des applications commerciales ou gouvernementales.

3.1.2. Caractéristiques d'un système expert

Le domaine des systèmes experts est le champ de l'intelligence artificielle qui vise à appliquer de la connaissance pour résoudre des problèmes qui requièrent habituellement l'intelligence humaine, dans un domaine bien précis.

Les systèmes experts mettent en oeuvre des méthodes symboliques, non algorithmiques de résolution de problèmes. En effet, une grande partie de la connaissance d'une personne ou d'un domaine comme la médecine, par exemple, n'est pas numérique ou quantitative. C'est une connaissance

symbolique, et elle est utilisée pour la résolution de problèmes. Les méthodes de résolution de problèmes elles-mêmes ne sont habituellement pas numériques mais des techniques de raisonnement qualitatif où interviennent des règles de jugement empiriques.

Les systèmes experts diffèrent des tâches classiques de l'intelligence artificielle par plusieurs caractéristiques. Premièrement, les systèmes experts exécutent des tâches complexes à un niveau de performance, que l'on espère égal à celui des experts. Ensuite, ils appliquent des stratégies de résolution de problèmes spécifiques au domaine plutôt que des méthodes générales de l'intelligence artificielle. Troisièmement, ils utilisent leur propre connaisance pour raisonner à propos de leur processus d'inférence et fournir des explications et des justifications aux solutions qu'ils proposent. Enfin, ils résolvent des problèmes qui appartiennent notamment aux catégories suivantes : interprétation, diagnostic, prévision, "design", planification, "monitoring", contrôle et enseignement.

Un système expert permet la résolution de problèmes complexes dans un domaine spécifique en utilisant une base de connaissances acquise auprès des experts du domaine et un mécanisme de raisonnement caractéristique de la façon de raisonner des experts. [PIN81]

3.1.3. Composants d'un système expert

Un système expert possède généralement :

- une base de connaissances spécifique du domaine
- une structure de contrôle qui, par un mécanisme de raisonnement, interprète la base de

connaissances et l'applique aux problèmes en vue de leur résolution

- un justificateur qui explique le comportement du système à l'utilisateur
- une interface permettant la communication entre le système expert et l'utilisateur ou l'expert dans un sous-ensemble du langage naturel ou au moyen de graphiques. D'une part, l'interface interprète les questions, les commandes et les informations de l'utilisateur. D'autre part, l'interface met en forme l'information générée par le système (réponses aux questions, explications et justifications de son comportement, demande de données).

3.1.4. La connaissance dans un système expert [HAY83]

Les systèmes experts résolvent des problèmes difficiles qui requièrent de l'expertise.

L'expertise est un ensemble de connaissances approfondies d'un domaine particulier, une compréhension des problèmes qui se posent dans ce domaine et les capacités de résoudre certains de ces problèmes.

La connaissance dans un système expert consiste en des données qui peuvent améliorer l'efficacité du système. On peut distinguer trois types de connaissances qui rentrent dans cette description :

- les "faits" constituent un corps de connaissances explicites, largement partagées, publiquement disponibles, et reconnues de manière générale par les experts d'un domaine. Ils expriment des expressions valides

- les "règles de déduction" permettent de déduire des connaissances implicites à partir des faits
- les "heuristiques" consistent essentiellement en des "règles de bonne pratique" (voies plausibles pour aborder un problème) acquises souvent grâce à une longue expérience. Ce sont des méthodes empiriques qui, lorsqu'elles sont appliquées à un problème donné, peuvent conduire à sa solution ou vers un progrès dans sa recherche, sans cependant donner la certitude de l'obtention d'une solution. Ces règles non précises sont introduites pour essayer d'utiliser efficacement les règles de déduction.

La puissance d'un programme intelligent à accomplir sa tâche dépend de la quantité et de la qualité de la connaissance qu'il possède sur sa tâche. La connaisance est l'ingrédient clé pour la résolution de problèmes complexes.

3.1.5. Le raisonnement dans un système expert

Souvent, les experts rencontrent des problèmes pour lesquels l'information est incomplète. Le système expert doit également effectuer un raisonnement dans un univers incertain et incomplet. Tout comme le médecin qui effectue un diagnostic, ou l'ingénieur qui conçoit un projet, il doit travailler avec une description partielle des situations.

Ces problèmes ne sont pas aisés à formaliser et ne possèdent pas de solution algorithmique. Contrairement aux applications algorithmiques traditionnelles, la plupart des systèmes experts travaillent dans des situations qui n'admettent pas de solutions que l'on puisse qualifier d'optimales. Dans de tels cas, le système doit trouver le meilleur compromis entre la qualité de la réponse qu'il

fournit et l'effort nécessaire à cette réponse. Dès lors, il doit mettre en oeuvre des méthodes heuristiques qui permettent d'utiliser efficacement la connaissance afin d'aller rapidement vers des voies prometteuses de solution.

Disposer de connaissances approfondies ne suffit pas dans la mesure où la performance de l'expert dépend également de son savoir-faire. Le système expert doit construire sa solution de manière sélective et efficace, développer une activité infructueuse aussi faible que possible, être robuste et commettre un minimum d'erreurs.

Un principe important dans l'organisation d'un système expert est de maintenir un raisonnement qui soit compréhensible pour le spécialiste du domaine. Ce principe n'est pas nécessaire d'un point de vue logique mais il facilitera notamment l'explication du raisonnement.

3.1.6. L'explication du raisonnement

Pour être acceptable, un système expert doit être capable de fournir des explications compréhensibles sur sa façon de raisonner : les actions qu'il a effectuées, les stratégies qu'il a employées et leur justification.

La capacité d'explication est un élément important d'un système expert pour plusieurs raisons : [BUC84]

- la compréhension du raisonnement : le concepteur du système et l'utilisateur ont besoin de comprendre le contenu de la base de connaissances et les lignes du raisonnement suivi afin de mettre à jour et d'utiliser efficacement le programme
- la mise au point du système expert : le processus d'explication permet au programmeur et à l'expert de

vérifier si tous les aspects du problème et toutes les règles à appliquer ont été pris en considération. En cas d'erreur ou d'incomplétude, ils peuvent, grâce à l'explication, localiser l'endroit où une modification s'avère utile

- l'apprentissage : afin de permettre à l'utilisateur d'apprendre quelque chose du système expert, il est nécessaire de rendre compréhensibles la base de connaissances et les lignes du raisonnement. L'utilisateur "naïf" peut alors enrichir ses connaissances en ce qui concerne le genre de problèmes traités par le système expert
- la persuasion de l'exactitude des résultats fournis par le système expert : il s'agit de convaincre les utilisateurs que les conclusions du système sont rationnelles. L'utilisateur doit comprendre suffisamment les conclusions que pour accepter la responsabilité d'agir en fonction de ces conclusions. Dans le domaine médical, par exemple, les médecins ont une responsabilité morale et légale des conséquences de leurs actions. Dès lors, ils doivent être persuadés que les conseils d'un expert sont appropriés.

Il est important que le système expert puisse éclairer l'utilisateur sur les points suivants :

- comment une décision a été prise
- comment une information a été utilisée
- pourquoi une information a été utilisée
- pourquoi une telle décision n'a pas été prise
- comment une certaine information a été découverte

Le système expert doit générer des explications en tenant compte de l'identité de l'utilisateur et de ses préférences. Selon que l'utilisateur est la personne

chargée de corriger le programme ou une personne "naïve" ne connaissant que quelques notions du domaine traité, le contenu et le niveau le détail de l'explication doivent être totalement différent.

En outre, afin que les explications soient compréhensibles, le système expert doit savoir, pour un même type d'utilisateur (par exemple l'utilisateur "naïf"), quel est le niveau de connaissance de l'utilisateur, ses limites et sa capacité à assimiler de nouvelles informations. Il doit donc pouvoir décrire la même information de différentes manières selon la personne à qui les explications sont destinées.

3.1.7. Types de systèmes experts [HAY83]

De nombreuses applications de manipulation de connaissances sont reprises dans les quelques types suivants. Les systèmes experts sont classés en fonction du type de tâche qu'ils prennent en charge :

3.1.7.1. Système d'interprétation

Il explique les données observées en leur assignant des significations décrivant la situation ou l'état du système. Cette catégorie inclut des activités de surveillance, compréhension de la parole, analyse de l'image, détermination de structures chimiques.

3.1.7.2. Système de diagnostic

Il infère des anomalies d'un système en se basant sur l'interprétation d'observations. Les systèmes de diagnostic

relient les irrégularités comportementales observées aux causes sous-jacentes. Les diagnostics médical, électronique, mécanique, parmi d'autres, sont concernés.

3.1.7.3. Système de prédiction

Il infère des conséquences probables à partir d'une situation donnée. Il emploie un modèle de valeurs paramétrées liées à la situation donnée. Cette catégorie inclut la prévision météorologique, la prévision démographique, l'estimation de récoltes, la prévision économique, entre autres.

3.1.7.4. Système de projet

Il développe des configurations d'objets qui satisfont les contraintes du problème. Cela inclut notamment le projet de construction, la conception de circuits, la conception de budgets.

3.1.7.5. Système de planification

Il crée des programmes d'actions qui peuvent être mis en oeuvre pour atteindre des objectifs. Cette catégorie inclut la planification d'expériences, de projets, les problèmes de planification militaire, la planification d'itinéraires.

3.1.7.6. Système de "monitoring"

Il interprète continuellement des signaux et déclenche des alarmes quand une intervention est nécessaire. C'est

notamment appliqué dans les centrales nucléaires, le trafic aérien, les hôpitaux.

3.1.7.7. Système de contrôle

Il dirige de manière adaptative tout le comportement d'un système. Pour faire cela, il doit, de manière répétée, interpréter la situation courante, prédire le futur, diagnostiquer les causes de problèmes anticipés, formuler un plan de remèdes et diriger son exécution. Les problèmes concernés sont le contrôle du trafic aérien, le "business management", le contrôle de mission.

3.1.7.8. Système d'enseignement [BON81]

Tel un professeur, il dispense un enseignement des connaissances d'un domaine. Il peut diriger l'étudiant par des stratégies pédagogiques qui tiennent compte de paramètres tels que les connaissances de l'étudiant, la vitesse d'apprentissage. Il peut détecter les fautes systématiques de l'étudiant et pas seulement répondre "vrai" ou "faux" à la solution apportée. Il peut générer des problèmes en fonction de paramètres décrivant la situation pédagogique, tels que le niveau de l'étudiant, accent à mettre sur un point précis. Il peut juger plusieurs méthodes et décider quelle est la meilleure.

3.1.8. Exemples

3.1.8.1. MYCIN [BUC84] [CLA87]

Il s'agit d'un système développé à partir de 1977 par Shortliffe et ses collègues. Il est destiné à établir un diagnostic et une thérapie dans le domaine des maladies infectieuses du sang, ou autre. Son travail consiste à récolter des renseignements, puis à donner un diagnostic sur la cause de l'infection (donner l'identité de la bactérie présente chez le patient), et à conseiller enfin une thérapie appropriée.

Divers projets ont été réalisés autour de MYCIN. Notamment, Guidon est un système expert permettant d'enseigner le diagnostic des maladies infectieuses à partir de la base de connaissances de MYCIN. GUIDON sélectionne un cas et le résout. Il le soumet ensuite à l'étudiant qui essaie de le résoudre. GUIDON analyse les réponses et les questions de l'étudiant durant la résolution du cas.

3.1.8.2. DENDRAL

Ce système expert fut développé à partir de la fin des années 60 principalement par Buchanan, Mithchell et Feigenbaum. Il s'agit d'un programme qui analyse des données provenant d'un spectographe de masse et donne des descriptions des structures de molécules qui, avec une très haute probabilité, donnent naissance à ces spectres. La compétence de ce programme est égale sinon supérieure à celle d'un chimiste humain dans l'analyse de certaines classes de molécules organiques.

3.1.8.3. MOLGEN [FEI79]

Ce système est le fruit d'une collaboration entre le "Genetic Department" et le "Heuristic Programming Project" de l'Université de Stanford. MOLGEN fournit un conseil intelligent à un généticien moléculaire sur la planification d'expériences impliquant la manipulation d'ADN. Certains programmes de MOLGEN offrent une assistance dans la planification, l'organisation et le séquencement d'expériences. D'autres permettent de vérifier si des plans d'expérience conçus par l'utilisateur sont réalisables.

3.1.8.4. PUFF [FEI79]

Ce système est né d'une collaboration entre le "Stanford Heuristic Programming Project" et le "Pacific Medical Center" de San Francisco. PUFF reçoit des données issues de tests sur les voies respiratoires d'un patient et certaines informations relatives au patient (âge, sexe, nombre de paquets de cigarettes fumés par an). Il infère à partir de ces données un diagnostic qui porte sur les points suivants : le patient est-il malade ?, la maladie éventuelle touche-t-elle les poumons ou les voies respiratoires ou les deux ?, quelle est la gravité de la maladie ?, quel est le type probable de maladie ?, etc.

3.2. LA CONTRIBUTION D'UN SYSTEME EXPERT A LA PURIFICATION DE SUBSTANCES D'ORIGINE BIOLOGIQUE

Après nous être familiarisés avec les connaissances de base du domaine des purifications, nous avons entrevu la possibilité de concevoir un système expert d'aide à la purification de substances d'origine biologique. Ce système

serait capable de supporter la tâche de conception d'une purification préalable à l'application des techniques de purification par le biochimiste. Nous nous sommes d'abord demandés si l'approche "système expert" était appropriée pour le problème.

Il n'est pas simple de décrire les caractéristiques qui rendent un problème approprié au développement d'un système expert. Il est plus simple de considérer la description d'un problème et d'évaluer ensuite s'il est approprié. Souvent, la bonne question n'est pas "un système expert peut-il apporter une solution à mon problème?", mais plutôt "quel aspect de mon problème se prête-t-il au développement d'un système expert ?". Ainsi, il n'est pas possible de rédiger une liste des problèmes appropriés. Par exemple, le domaine de la médecine n'est pas plus approprié que la chimie ou la géologie. Mais le problème du diagnostic et de la prescription de thérapie tel qu'on le rencontre en médecine se prête particulièrement bien au développement d'un système expert.

Cependant, une organisation qui pense à développer un système expert a une perspective assez différente. Sa première question sera "un système expert peut-il résoudre mon problème ?". Bien qu'on ne puisse répondre de manière simple et générale à cette question, des lignes de conduite, permettant de mener une réflexion, existent [WAT86]. Avant même de réaliser une description précise du problème, nous avons essayé de les suivre pour relever les caractéristiques du problème qui peuvent justifier l'approche "système expert".

Tout d'abord, nous devions pouvoir nous adresser à de véritables experts du domaine, possédant un niveau élevé d'expertise dans le domaine et capables d'expliquer les méthodes appliquées pour résoudre les problèmes de leur domaine. Les biochimistes du laboratoire réalisent souvent des purifications et disposent d'un important matériel supportant un grand nombre de techniques de purification. Ils connaissent bien ces techniques et les enseignent aux étudiants de biologie lors de cours et de laboratoires.

La tâche à résoudre doit exiger des capacités intellectuelles. Bien que la purification comporte des activités exclusivement manuelles qui ne peuvent être enseignées que par la pratique, la tâche requiert une combinaison de connaissances intellectuelles et d'adresse manuelle. Les activités intellectuelles de la tâche peuvent être prises en charge par un système expert.

La tâche ne doit pas être d'une difficulté extrême. Si un expert ne peut expliquer le processus parce que l'expertise ne peut être acquise que par une expérience de travail, le processus sera difficile à introduire dans un système expert. En effet, les experts ne pourront décrire leur expertise aux informaticiens. Si un expert prend des jours plutôt que des heures pour résoudre un problème, il y a de fortes chances pour qu'une approche "système expert" soit trop complexe. Le problème de la purification ne semble pas trop difficile pour l'aborder avec une approche "système expert" parce qu'il est possible de décomposer le problème en différentes sous-tâches. Dans une première approche, une technique de purification peut être abordée indépendamment des autres techniques, avant d'être incorporée dans une approche globale du problème. Un prototype du système expert peut prendre en charge la résolution du problème sur base d'un nombre limité de techniques de purification.

La tâche doit appartenir à un domaine que les experts maîtrisent bien. Les experts doivent comprendre parfaitement le domaine, la connaissance doit être bien structurée. Si la tâche est si récente qu'elle n'est pas bien comprise, un système expert n'est pas adapté. Comme nous l'avons déjà signalé, le problème des purifications existe depuis le début du développement de la biochimie. Le problème est bien connu. Les recherches sont nombreuses. Les techniques de purification connaissent sans cesse des améliorations. La difficulté de la tâche vient de la quantité de connaissances à maîtriser pour atteindre un haut niveau d'expertise. Toutefois, cela ne semble pas rendre impossible la maîtrise du domaine.

Pour justifier la réalisation d'un système expert, on peut penser à différents éléments. Une entreprise peut jutifier le développement d'un système expert si celui-ci est à même de procurer un profit important. Si les experts du domaine ne sont pas que rarement disponibles pour résoudre les problèmes posés, le développement peut se justifier. Un système expert peut être utilisé à différents endroits d'un laboratoire ou d'une entreprise, contrairement à l'expert humain. Puisqu'il est reproductible, le système expert peut être une solution économique. Un sytème expert peut se justifier si l'expertise risque d'être perdue par le départ des experts humains. Par exemple, dans une organisation quelconque, il y a des départs en retraite, des mutations, etc. La perte de l'expertise au travers du départ des personnes expérimentées peut être fatale à certaines activités de l'organisation. Un système expert peut s'avérer utile pour réaliser une tâche dans un environnement hostile et dangereux comme une centrale nucléaire par exemple. Dans le problème de la purification, il semble que la justification essentielle est la distribution d'une expertise rare à des personnes

travaillant dans le domaine, mais n'ayant pas acquis un niveau élevé d'expertise. Un premier public intéressé par un tel système pourrait être les chercheurs et les opérateurs de laboratoires. Ils recherchent de l'expertise pour réaliser une purification. D'une part, ils pourraient trouver un conseil intéressant auprès du système, concevant une purification adaptée aux possibilités de leur laboratoire et aux exigences qu'ils veulent imposer à la purification. D'autre part, le système pourrait leur permettre d'acquérir une certaine expertise, dans la mesure où le système assure une explication du raisonnement réalisé. Cet outil pourrait s'avérer utile pour les étudiants. Il est certain que le conseil fourni par le système sera plus performant pour des cas de purification sur lesquels les connaissances sont importantes. On rencontre cette condition pour les "cas d'école" que l'on soumet aux étudiants lors de l'étude des techniques de purification. On peut donc raisonnablement penser que le système pourra être adapté en outil d'enseignement.

La nature du problème de purification est adaptée à la conception d'un système expert. Il n'admet pas de solution exclusivement numérique, mais requiert un raisonnement symbolique. La recherche d'une solution à la purification d'une substance particulière n'admet pas de solution que l'on peut qualifier de correcte. On utilise des règles permettant d'atteindre une purification acceptable.

Le problème est complexe. Il exige des connaissances étendues à propos des techniques de purification et des mélanges biochimiques pour atteindre le niveau d'expert. Si la tâche était trop simple, un système expert ne se justifierait pas parce qu'il demanderait de mettre en oeuvre trop de moyens pour l'importance du résultat.

L'étendue du problème doit être choisie de manière adéquate. Le problème doit être suffisamment étendu que pour avoir une valeur pratique aux yeux des personnes travaillant dans le domaine, mais rester suffisamment étroit que pour être manipulable. Le choix de l'étendue du problème est crucial. L'une des erreurs les plus néfastes au succès de la conception du système expert est de choisir un problème trop large et trop général. Puisque nous pouvons aborder le problème de la purification en concevant un prototype ne prenant en charge que quelques techniques de purification, nous pouvons partager le problème de manière à le rendre manipulable. Nous l'avons vu au chapitre 2 que les biochimistes ne disposent pas toujours d'une aide suffisante pour réaliser les purifications nécessaires à leur travail. Leurs connaissances sont parfois trop restreintes et les purifications présentées dans la littérature sont souvent inadaptées à leurs besoins. Comme nous l'avons expliqué ci-dessus, la conception du système présente une valeur à leurs yeux.

PARTIE 2

CADRE METHODOLOGIQUE

CHAPITRE 4. UNE METHODE DE CONSTRUCTION D'UN SYSTEME EXPERT

Bien qu'il n'existe pas encore de méthode de construction de système expert qui soit universellement acceptée, les spécialistes du domaine s'accordent généralement sur une succession d'étapes que nous avons essayé de suivre pour la construction du système de purification. Nous avons tenté de suivre les propositions de [BUC83], [WAT86], [GAN85].

Le processus de construction d'un système expert est souvent appelé le "knowledge engineering". Il implique un travail d'équipe d'un concepteur de système expert, appelé l'ingénieur de connaissance, et d'un ou de plusieurs experts humains du domaine. L'ingénieur de connaissance "extrait" des experts humains leurs connaissances pour résoudre les problèmes du domaine. Ce processus d'acquisition des connaissances consiste dans le transfert et la transformation de l'expertise pour la résolution d'un problème depuis une source de connaissances jusqu'à un programme. L'expertise est un ensemble de faits, de procédures et de règles de jugement à propos d'un domaine étroit plutôt qu'une connaissance générale d'un domaine. L'"extraction" des connaissances auprès des experts du domaine est un problème important et difficile.

Un système expert est construit par un processus de développement itératif. L'acquisition des connaissances n'est pas directe et définitive. Les retours en arrière peuvent être nombreux pour améliorer la qualité et la quantité des connaissances. Après une conception et une implémentation initiales, le système croît. Ce processus itératif est indispensable à la conception d'un tel système.

Nous présentons une démarche très générale. Nous identifions les différentes étapes du processus. Elles ne suivent pas un ordre chronologique parfait. Au contraire, elles se recouvrent souvent.

4.1. L'ETAPE D'IDENTIFICATION

La première étape dans le processus d'acquisition des connaissances est de caractériser les aspects importants qui permettront le développement du système expert : déterminer les participants au développement, définir les caractéristiques du problème, évaluer les ressources nécessaires, déterminer les objectifs du système.

4.1.1. Identification des participants et de leur rôle

L'équipe de développement du système expert peut être composée d'experts du domaine, d'ingénieurs de connaissance et de spécialistes interdisciplinaires.

Le choix des experts est important. En effet, c'est d'eux que provient la plus grande partie de la connaissance, et il importe de les choisir de façon à obtenir l'expertise la plus complète possible. C'est ainsi qu'on peut parfois être amené à un choix pluridisciplinaire.

Dans les premiers mois du développement, il est conseillé de recourir à un petit nombre d'experts du domaine afin d'établir plus facilement un consensus. Par la suite, il sera intéressant de soumettre les connaissances déjà acquises à d'autres experts, de recouper les renseignesments provenant de ces experts, de confronter chacun avec le

résultat de ce recoupement, pour obtenir une connaissance plus fine et plus précise.

L'expert doit être une personne qui connaît le domaine pour l'avoir pratiqué pendant plusieurs années. Il ne lui suffit pas d'avoir une compréhension théorique du domaine, mais il doit également avoir une grande expérience pratique. Le rôle de l'expert est de transférer vers l'ingénieur de connaissance la connaissance nécessaire à la construction du système.

Puisque les ingénieurs de connaissance ont peu de connaissances du domaine, leur rôle se limite, dans un premier temps, à se familiariser avec le domaine et les problèmes qu'on y rencontre. Les ingénieurs de connaissance doivent être capables d'assimiler une grande partie des connaissances de base du domaine. Lors des premières rencontres entre experts et ingénieurs, le vocabulaire utilisé pour parler du domaine est simple et inapproprié pour la résolution de problèmes. Le vocabulaire est ensuite étendu et raffiné. Le rôle des ingénieurs de connaissance est alors de "pousser" les experts à donner leur connaissance et à formuler explicitement les mécanismes qu'ils utilisent pour résoudre leurs problèmes. Il faut aussi essayer de leur "extraire" la connaissance qu'ils ont accumulée durant leur expérience. Cette "extraction" des connaissances n'est pas un problème simple. Les experts du domaine ont en général de grandes difficultés à exprimer leurs connaissances et le raisonnement qu'ils suivent pour résoudre un problème. En général, ils réalisent des raisonnements complexes rapidement sans en examiner avec précision les étapes. Ils disposent d'éléments de connaissance qu'ils combinent si rapidement qu'il devient difficile pour eux de décrire le processus de raisonnement. Parfois ils n'ont plus conscience d'une étape de base du raisonnement. Ils se basent aussi sur l'intuition.

4.1.2. Identification du problème

Lorsque les participants ont été choisis, l'ingénieur de connaissance et l'expert du domaine peuvent procéder à l'identification du problème. L'objectif est de fournir une description du problème pour que le développement d'une base de connaissances puisse commencer. Cette description cernera le problème et les structures de connaissances pertinentes pour la résolution du problème. Cela nécessite des échanges de vues informels où les questions suivantes seront abordées :

- Quelle classe de problèmes le système expert devra-t-il résoudre ?
- Comment ces problèmes peuvent-ils être définis ?
- Quels sont les sous-problèmes importants ? Quelle est la répartition des tâches qui en découle ?
- Quelles sont les données ?
- Quels sont les concepts-clés importants et leurs relations ?
- A quoi ressemble une solution et quels sont les concepts qu'elle fait intervenir ?
- Quels aspects de l'expertise humaine sont essentiels dans la résolution de ces problèmes ?
- Quelles explications peuvent être fournies avec une solution ?
- Quelles sont les situations qui semblent empêcher une solution ?
- Quelles en seront les répercussions sur le système expert ?

L'identification du problème, de son étendue n'est pas toujours aisée. L'équipe de développement doit peut-être envisager l'identification du problème à plusieurs reprises.

L'expert décrit des problèmes typiques, explique leur résolution et le raisonnement qui sous-tend cette résolution. Après plusieurs tentatives, une formulation définitive du problème est dégagée.

Le problème doit être circoncris de manière précise dans le domaine de l'expert.

4.1.3. Identification des ressources

Des ressources typiques sont les sources de connaissance, le temps, les moyens informatiques et l'argent. Elles sont nécessaires pour acquérir la connaissance, l'implémenter et la tester.

L'expert du domaine puisera des connaissances dans son expérience personnelle, la littérature du domaine, des études de cas, des bases de données.

L'ingénieur de connaissance se référera à des problèmes analogues traités dans le domaine des systèmes experts. Il y découvrira des méthodes, des représentations, des outils propres à ce domaine.

Le temps est une ressource critique. L'ingénieur de connaissance et l'expert du domaine doivent consacrer plusieurs mois d'activité pour permettre la construction d'un premier prototype.

Les moyens informatiques nécessaires doivent rester disponibles tout au long du processus de développement. Il doit s'agir aussi bien de logiciels, outils appropriés pour la construction des systèmes experts, que de matériels.

4.1.4. Identification des objectifs

Il est probable que l'expert définira les objectifs du système en même temps que d'identifier le problème. Cependant, il est utile de saisir la différence des points de vue. Décrire les objectifs du système, c'est décrire l'utilité du système dans l'endroit où il sera implanté. Des exemples d'objectifs peuvent être la formalisation d'un ensemble de pratiques informelles, la distribution d'une expertise rare, l'amélioration des méthodes de résolution de l'expert, l'automatisation des aspects routiniers du travail de l'expert.

4.2. L'ETAPE DE CONCEPTUALISATION

L'objectif de cette étape est de décrire les concepts-clés et leurs relations, les sous-problèmes dégagés lors de l'étape d'identification du problème. L'ingénieur de connaissance et l'expert examineront également des concepts et relations secondaires.

Cette étape doit prendre la forme d'un processus interactif entre les participants au développement. Les questions abordées seront les suivantes :

- Quels types de données sont disponibles ?
- Qu'est-ce qui est donné et qu'est-ce qui est déduit?
- Les sous-tâches ont-elles un nom ?
- Les stratégies ont-elles un nom ?
- Quelles sont les hypothèses généralement utilisées ?
- Comment les objets sont-ils reliés dans le domaine ?

- Peut-on réaliser un diagramme de la hiérarchie des sous-tâches et identifier des relations telles que les relations causales, des inclusions, ou d'autres relations bien définies ?
- Quels sont les traitements qui interviennent dans la résolution du problème ?
- Quelles sont les contraintes sur ces traitements ?
- Quel est le flux d'information ?
- Comment peut-on identifier et séparer la connaissance nécessaire pour résoudre un problème et la connaissance utilisée pour justifier une solution ?

Lors des entretiens, différents moyens peuvent être utilisés : dessins, graphiques, organigrammes, etc. L'ingénieur de connaissance peut trouver utile de réaliser des diagrammes représentant la structure des étapes du raisonnement de l'expert. Ces diagrammes pourront servir lors de phases ultérieures. Les hésitations, les retours en arrière de l'expert, en un mot tous les évènements qui n'ont pas leur place dans une présentation académique sont importants à noter. Ils peuvent cacher des connaissances.

L'ingénieur peut commencer à décrire une architecture globale pour la base de connaissances et les mécanismes de contrôle qui viendront se greffer sur le raisonnement de l'expert.

L'expérience a montré qu'il ne faut pas essayer d'analyser complètement et correctement le problème avant d'implémenter un prototype. Cela provient peut-être du fait, qu'habituellement, pour traiter un problème délicat, l'homme procède par étapes : dans un premier temps, il fait un raisonnement général pour cerner les difficultés. Puis, il affine progressivement son raisonnement jusqu'à ce que plus aucune contradiction n'apparaisse. Aussi, les

concepts-clés et les relations pourront être précisés s'ils sont travaillés lors d'étapes de formalisation et d'implémentation.

L'ingénieur de connaissance peut penser à des représentations de la connaissance et à des outils afin de diriger la conceptualisation, mais il ne doit pas sélectionner en particulier l'une ou l'autre solution.

4.3. L'ETAPE DE FORMALISATION

Le processus de formalisation consiste à exprimer les concepts-clés, les sous-problèmes et les caractéristiques du flux d'information dégagés lors de l'étape de conceptualisation dans des représentations plus formelles. L'ingénieur de connaissance prend alors un rôle plus actif. Il envisage les outils de développement, les représentations de connaissances que ces outils favorisent, les types de problèmes qui semblent correspondre au problème à résoudre.

Il étudie la structure des concepts : est-ce utile de décrire les concepts comme des objets structurés ou comme des entités simples ? Il détermine la représentation des relations entre ces concepts : y-a-t-il des relations causales ou spatio-temporelles entre les concepts et peut-on les représenter explicitement ? Cette étude fournit des indications sur la structure de l'espace de recherche des solutions.

L'ingénieur de connaissance doit étudier la nature des données : sont-elles clairsemées ou abondantes ? Sont-elles attachées à un degré d'incertitude ? Quel est le coût de l'acquisition des données ? Comment peut-on les acquérir ? Quel type de questions peut-on poser pour les obtenir ? Les

données sont-elles cohérentes pour les problèmes qui ont été résolus ?

Découvrir un modèle du processus utilisé pour résoudre les problèmes dans le domaine est une étape importante dans la formalisation des connaissances. Ce modèle doit intégrer les connaissances. Le modèle sert à justifier la cohérence des concepts et des relations entre ceux-ci.

L'ingénieur de connaissance détermine la représentation des connaissances, les langages et les outils qui lui semblent convenir pour le problème à résoudre. Il les présente à l'expert humain. Ensemble, ils font un choix.

Le résultat de cette étape est un ensemble de spécifications partielles décrivant comment le problème peut être représenté avec les outils choisis. Elles peuvent servir pour la construction de la base de connaissances d'un prototype du système. Ces spécifications précisent le contenu des structures de données, des règles d'inférence et des stratégies de contrôle.

Il est possible que différents langages et outils de construction de systèmes experts soient adaptés pour des sous-problèmes différents. L'ingénieur de connaissance peut décider, dans ce cas, d'utiliser plusieurs outils. Cependant, il doit alors évaluer les désavantages occasionnés. Il doit intégrer ces outils les uns aux autres. Souvent, les langages et les outils sont choisis en fonction du sous-problème le plus important.

4.4. L'ETAPE D'IMPLEMENTATION

La connaissance formalisée doit à présent être utilisée dans un programme. La forme de la connaissance est spécifiée par les formes de représentation offertes par l'outil ou le langage choisi. L'ingénieur de connaissance doit intégrer et organiser les différentes pièces de la connaissance pour définir une structure de contrôle et un flux d'information qui forment un prototype de programme.

L'ingénieur de connaissance utilise le formalisme choisi pour capter les concepts, les processus d'inférence et la stratégie de résolution du problème utilisée par l'expert. Il peut déceler des incohérences introduites lors d'étapes précédentes. Il doit éliminer les contradictions dans les structures de données et les règles.

Une partie du code écrit pour ce prototype sera repris dans des versions ultérieures du système. Mais le travail le plus important est de vérifier la cohérence de la formalisation, l'efficacité des décisions de conception réalisées durant les premières étapes de développement. La probabilité est grande de voir le code initial modifié à de multiples reprises.

Le résultat de cette étape est un programme exécutable, prototype du système expert définitif.

4.5. L'ETAPE DE TEST

Cette étape a pour objectif l'évaluation de la performance du prototype et des formes de représentation utilisées pour l'implémenter. Lorsque le système tourne

complètement sur deux ou trois exemples, il peut être testé sur une variété d'exemples destinés à montrer les faiblesses dans la base de connaissances et la structure d'inférence. Cette évaluation peut faire apparaître des lacunes comme le manque de concepts et de relations, une connaissance représentée à un mauvais niveau de détail. Les éléments qui sont habituellement la cause de faibles performances sont les entrées/sorties, les règles d'inférence, les stratégies de contrôle, les exemples de tests, etc.

Les caractéristiques principales des entrées/sorties sont l'acquisition des données et la présentation des conclusions.

La qualité de l'acquisition des données auprès de l'utilisateur influence bien sûr la qualité de la performance du système. La méthode employée pour acquérir les données peut être inadéquate. Par exemple, les questions peuvent être mal posées, difficiles à comprendre, ou ambiguës pour l'utilisateur du système. Ainsi l'information récoltée peut être insuffisante ou incohérente pour la résolution du problème.

Les conclusions apportées par le système doivent être adéquates et organisées. Le niveau de détail est important : les conclusions ne doivent pas être trop ou pas assez détaillées. L'ordre de présentation des conclusions à l'utilisateur est également important.

Les erreurs de raisonnement du système se logent souvent dans les règles d'inférence. Elles sont rarement indépendantes l'une de l'autre. Il faut vérifier l'existence, la cohérence, la compatibilité et la complétude des règles. Une règle peut être appliquée dans un contexte incorrect. Les conclusions apportées par une règle peuvent être incorrectes. Les conditions d'application et les

conclusions d'une règle peuvent être associées de manière incorrecte. Lorsque les règles ne sont pas indépendantes, leurs effets sur d'autres règles doivent être considérés.

Des erreurs peuvent se glisser également dans les stratégies de contrôle. Lorque le système considère des éléments dans un ordre que l'expert ne trouve pas naturel, l'ingénieur de connaissance doit regarder les stratégies de contrôle.

Le choix des exemples de tests est important également. Des erreurs peuvent être découvertes grâce à un choix judicieux des exemples. L'ingénieur de connaissance et l'expert doivent construire des jeux de tests qui permettent de couvrir toutes les classes de sous-problèmes, aux limites de ces classes.

On peut également penser, à ce stade, à évaluer l'utilité du système. On se demande si la solution au problème, telle qu'elle est proposée par le système, aide l'utilisateur de manière significative.

4.6. LA REVISION DU PROTOTYPE

Dans la construction d'un système expert, il y a presque constamment des révisions : reformulation de concepts, reconception de représentations ou raffinement du système. Le raffinement concerne des règles d'inférence et des structures de contrôle jusqu'au moment où on obtient le comportement attendu.

L'objectif est essentiellement l'amélioration des performances du système. Il peut s'avérer intéressant d'augmenter les sources de connaissances afin d'enrichir la

base de connaissances. Lorsqu'on dispose d'un prototype, on peut le soumettre à des experts du domaine n'ayant pas participé à son développement. En utilisant les capacités "auto-explicatives" du système, ces experts externes peuvent en observer le comportement, le critiquer, voire apporter de nouvelles connaissances.

Le système expert doit d'abord être raffiné et testé dans un environnement de laboratoire. Lorsqu'il est soumis à l'utilisateur et placé face à de réels problèmes, de nouvelles complications apparaissent. Leur correction peut être relativement longue. De plus, les utilisateurs demandent davantage que la performance de haute-qualité. Ils veulent un système qui soit rapide, sûr, facile à utiliser, facile à comprendre et indulgent lorsqu'ils font des erreurs.

CHAPITRE 5. OUTILS ET LANGAGES

Comme nous l'avons vu au chapitre précédent, la conception d'un système expert est un processus évolutif et complexe, qui requiert beaucoup de temps. Le coût de développement est élevé. Aussi, les chercheurs en intelligence artificielle mettent-ils au point des outils destinés à assister les concepteurs d'un système expert. Certains systèmes intègrent également des outils qui s'adressent directement à l'utilisateur du système expert.

Bien que les chercheurs essaient de concevoir des outils généraux, permettant de prendre en charge différents types de problèmes dans des domaines divers, les outils les plus performants sont adaptés plus particulièrement à un type de problèmes, un type de conception, un type d'implémentation. Un choix approprié d'outil permet de réduire le temps de développement du système expert de manière importante et de construire un produit qui atteint un haut degré de performance.

Les outils peuvent supporter une ou plusieurs méthodes de représentation des connaissances. Nous pensons aux représentations souvent utilisées comme les règles, les frames et les réseaux sémantiques, mais aussi les représentations "orientées objets" et les représentations clausales basées sur la logique des propositions. Notre objectif n'est pas, dans ce chapitre de développer ces méthodes de représentation, mais de présenter brièvement différents types d'outils classés en fonction de leur niveau de sophistication. Les outils présentés utilisent, pour la plupart, une représentation des connaissances sous forme de règles.

5.1. LES LANGAGES DE PROGRAMMATION

Divers langages de programmation ont été utilisés pour développer des systèmes experts : des langages plus spécifiques à des applications algorithmiques comme FORTRAN et PASCAL et des langages de "manipulation de symboles" comme LISP et PROLOG. FORTRAN est souvent utilisé pour résoudre les problèmes posés, notamment, dans des domaines comme les mathématiques et les statistiques. Les langages "de manipulation de symboles" ont été conçus pour développer des applications en intelligence artificielle. Bien que quelques systèmes experts aient été écrits dans des langages spécifiques aux applications algorithmiques, ils sont moins adéquats pour travailler en intelligence artificielle. Les programmes écrits dans ces langages mélangent les données et le contrôle. Les connaissances sont diluées dans le corps des procédures et des représentations utilisées. Pour construire un système expert, il est plus adéquat d'entrer les connaissances par fragments compacts et indépendants sans préjuger de la manière dont ceux-ci seront utilisés.

Le langage le plus répandu pour les applications de systèmes experts est LISP, bien que PROLOG gagne en popularité. LISP comprend des mécanismes pour la manipulation des symboles sous la forme de structures de listes. Une liste est un ensemble ordonné d'objets entourés de parenthèses dans laquelle chaque objet peut être un symbole ou une autre liste. Les structures de listes sont utiles pour construire des agrégats représentant des concepts complexes. Une relation entre des objets est souvent représentée par une liste contenant la relation suivie des objets reliés. La manipulation des symboles est facile et flexible. Des outils d'édition et de "debugging" sophistiqués sont souvent incorporés. LISP réalise un traitement uniforme du code du programme et des données,

c'est-à-dire que LISP peut modifier son propre code, aussi facilement que des données. Pour développer un système expert dans lequel la représentation sous forme de règles est utilisée, deux composants importants doivent être développés : un moteur d'inférence et un ensemble de règles. Le concepteur d'un système expert doit développer un langage et un ensemble de concepts dans lesquels les règles puissent être exprimées. Puis il faut concevoir le moteur d'inférence qui exploite les règles. En LISP, des techniques ont été développées pour implémenter de tels composants d'un système expert.

Les langages comme LISP offrent la flexibilité pour la représentation des concepts d'une application de système expert, mais ne procurent pas d'aide sur la manière de représenter la connaissance d'un domaine, ni sur les mécanismes pour accéder à la base de connaissances.

5.2. LES LANGAGES DE "KNOWLEDGE ENGINEERING"

Ces outils forment une classe particulière des langages de programmation. Ils ont été conçus expressément pour construire des systèmes experts. Ils comprennent un langage de construction de système expert intégré dans un environnement. Beaucoup de ces systèmes ont été conçus en LISP. Nous pouvons classer ces outils en deux groupes : les "skeletal" systèmes et les "general-purpose" systèmes.

5.2.1. Les "skeletal" systèmes

Un "skeletal" système est construit à partir d'un système expert. Dans ce système expert, la connaissance a été représentée explicitement en dehors du moteur

d'inférence. Dès lors, il est possible d'enlever les connaissances d'un domaine et de les remplacer par des connaissances assurant la résolution d'une autre tâche. Un tel processus simplifie bien sûr la construction du système expert prenant en charge la seconde tâche.

Cependant, en pratique, ce processus se déroule rarament aussi facilement. Le système, vidé de sa base de connaissances peut être inapproprié pour la nouvelle tâche. C'est le problème de plus fréquent et le plus sérieux. La structure de contrôle incluse dans le moteur d'inférence peut être insuffisante pour prendre en charge la résolution des problèmes posés par la nouvelle tâche. Le langage d'expression des règles peut être inapproprié. Si ces difficultés peuvent être contournées, l'utilisation d'un "skeletal" système peut s'avérer efficace, permettant de construire un système expert assez rapidement.

Les concepteurs du système expert MYCIN ont enlevé la connaissance qu'il contenait dans le domaine des maladies infectieuses du sang. Ils ont ainsi obtenu EMYCIN, un système adapté pour le développement de systèmes experts de diagnostic. Le système expert PUFF a été construit en utilisant EMYCIN. Le sytème expert CASNET a donné naissance au "skeletal" système EXPERT.

L'avantage essentiel de tels systèmes de développement est que tout outil incorporé dans le système, comme un mécanisme d'explication du raisonnement par exemple, est disponible pour le nouveau système expert. Les inconvénients sont qu'ils manquent de flexibilité et de généralité en ce qui concerne la représentation et la manipulation des connaissances.

5.2.2. Les "general purpose" systèmes

Certains systèmes experts ne peuvent être construits sur un "skeletal" système. Aussi, des "general-purpose" systèmes ont été conçus. Ils peuvent manipuler des problèmes de types différents. Ils fournissent au concepteur du système expert davantage de possibilités qu'un "skeletal" système en ce qui concerne l'accès et la recherche des données. Ils sont moins liés à une architecture ou un paradigme spécifique. Cependant, ils sont souvent plus difficiles à utiliser.

OPS5 est un outil qui entre dans cette catégorie. OPS5 incorpore des mécanismes de représentation et de contrôle généraux. Il fournit les mécanismes de base pour le "knowledge engineering", mais ne comprend pas de schémas particuliers de représentation ou de résolution de problèmes. OPS5 permet d'utiliser des symboles et d'exprimer des relations entre les symboles, mais aucun symbole et aucune relation n'ont reçu de signification prédéfinie. Les significations sont entièrement décrites par les règles de production écrites par le programmeur. Le mécanisme de contrôle de l'interpréteur d'OPS5 est un cycle, appelé le cycle "recognize-act", que l'ingénieur de connaissances utilise comme il le désire. Plutôt que de changer l'interpréteur, l'ingénieur de connaissances peut écrire des règles pour effectuer le contrôle. Ces règles sont manipulées comme un type de connaissances sur lesquelles l'interpréteur raisonne.

5.3. LES OUTILS D'AIDE A LA CONSTRUCTION

Ces outils sont des environnements qui aident le concepteur d'un système expert à acquérir et à représenter

la connaissance du domaine de l'expert, à construire le système expert. Ils sont adaptés lorsqu'un concepteur se trouve face à une tâche d'une difficulté importante. Par rapport aux langages de programmation et aux langages de "knowledge engineering", peu de systèmes d'aide à la construction ont été conçus. Ceux qui existent se répartissent en deux catégories : les aides à la conception et les aides à l'acquisition de connaissances. Nous allons illustrer chaque catégorie par un exemple : AGE est un système d'aide à la conception et TEIRESIAS est un système d'aide à l'acquisition de connaissances.

5.3.1. AGE [BAR83]

AGE aide l'ingénieur de connaissances à construire un système expert. Il fournit une aide pour concevoir une architecture, concevoir un langage d'expression des règles. L'objectif de cet outil est de permettre à l'ingénieur de connaissances de choisir ou de spécifier différentes représentations de la connaissance et différentes méthodes de résolution.

AGE fournit à l'ingénieur de connaissances un ensemble de composants qui, comme des briques de construction, peuvent être assemblées pour former des parties du système expert. Chaque composant est une collection de fonctions INTERLISP et supporte une partie du système expert. Par exemple, le composant "production-rule" est composé d'un interpréteur de règles qui accepte les règles de production et des stratégies pour la sélection et l'exécution des règles. L'ingénieur de connaissances intègre un composant dans son système expert si les fonctions assurées par ce composant sont utiles à la résolution de la tâche prise en charge par le système expert. Les composants peuvent être modifiés pour autant que les changements apportés restent

conformes aux contraintes imposées par la définition du composant.

Les composants de AGE ont été définis et programmés pour être utilisables dans de multiples combinaisons. Cela permet de construire des programmes basés sur des comportements différents de résolution de problèmes . Pour les ingénieurs de connaissances peu habitués à combiner les composants, AGE inclut des configurations prédéfinies de composants. Chaque configuration est appelée un "framework". Pour chacune, les décisions de conception de base ont été prises, mais les décisions finales restent à réaliser par l'ingénieur de connaissances en fonction de ses besoins et de ses préférences.

Age a été conçu pour un ingénieur de connaissances expérimenté ayant une solide connaissance du langage INTERLISP, un des dialectes les plus connus du LISP.

5.3.2. TEIRESIAS [DAV81]

TEIRESIAS aide au transfert de connaissances d'un domaine dans une base de connaissances. Le système acquiert de nouvelles règles sur un problème du domaine au moyen d'une interaction avec des utilisateurs qui établissent des règles dans un sous-ensemble de l'Anglais. Le système analyse les règles, fait des suggestions en considérant leur complétude et leur cohérence, aide l'utilisateur à les corriger lui-même. Un outil tel que TEIRESIAS peut être utilisé non seulement par les concepteurs du système expert, mais aussi par des utilisateurs "naïfs".

5.4. LES ENVIRONNEMENTS DE DEVELOPPEMENT DE SYSTEMES EXPERTS

Les environnements de développement de systèmes experts regroupent des outils pour aider à la programmation, à l'édition de base de connaissances, au "debugging", augmentent les capacités du système construit au moyen de procédures d'"entrées/sorties" et de mécanismes d'explication prédéfinis. Ces environnements sont habituellement construits autour d'un langage de "knowledge engineering" et sont conçus pour travailler spécifiquement avec ce langage. Un tel environnement regroupe un ensemble de couches qui viennent s'ajouter au langage de "knowledge engineering" pour rendre son utilisation plus facile, plus conviviale et plus efficace.

Nous allons décrire quatre composants typiques d'un environnement de développement de système expert : l'aide au "debugging", les procédures prédéfinies d'"entrées/sorties", les mécanismes d'explication et les éditeurs de bases de connaissances. Tous les environnements ne comprennent pas ces composants.

5.4.1. Les outils de "debugging"

Les outils de "debugging", que comportent beaucoup de langages de "knowledge engineering", comprennent des possibilités de "tracing" et de "breaking". Le "tracing" permet à l'utilisateur de suivre le déroulement du programme qu'il met au point. Ce mécanisme affiche les règles appliquées et les fonctions exécutées, ainsi que le contenu des variables utilisées. Le "breaking" permet au concepteur de dire, à l'avance, où l'exécution du programme doit s'arrêter afin d'examiner l'état de la base de connaissances

et des variables du programme en général, et ainsi repérer des erreurs éventuelles.

Quelques outils de "debugging" incorporent également une possibilité d'"automated testing". Cela permet de soumettre au programme un grand nombre de problèmes pour les tester automatiquement et découvrir les erreurs et les incohérences dans les solutions. Par exemple, EXPERT peut mémoriser des centaines de cas de tests avec leur solution correcte et les utiliser pour tester automatiquement les règles dans la base de connaissances. Les concepteurs d'un système expert peuvent trouver l'"automated testing" particulièrement utile lorsqu'ils réalisent une révision ou une extension du système expert, car ils peuvent rapidement tester si les extensions ont introduit des erreurs dans la base de connaissances.

5.4.2. Les interfaces

Les outils manipulent les "entrées-sorties" de différentes manières. Certains comportent des mécanismes qui permettent à l'utilisateur de converser directement avec le système expert. Par exemple, les systèmes experts développés sur EMYCIN demandent à l'utilisateur des informations lorsqu'ils ne peuvent les trouver dans la base de connaissances. Sur EXPERT, non seulement le système demande des informations manquantes à l'utilisateur, mais présente des menus dans lesquels l'utilisateur peut sélectionner l'information. L'utilisateur peut également donner volontairement des informations pendant l'exécution du système expert. Lorsqu'un ingénieur de connaissances utilise EMYCIN ou EXPERT, il ne doit pas écrire de code de programme pour l'acquisition des connaissances. De tels outils facilitent le développement du système expert, mais la flexibilité est réduite. En effet, les "entrées/sorties"

sont manipulées de la manière conçue par le concepteur de l'outil.

Les outils de développement de systèmes experts peuvent aussi fournir un ensemble de primitives puissantes qui rendent la programmation des procédures d'"entrées/sorties" assez facile.

5.4.3. Les mécanismes d'explication

C'est une nécessité pour un système expert d'expliquer aux concepteurs et aux utilisateurs surtout, le raisonnement développé pour atteindre les conclusions et la solution qu'il obtient. Aussi, Il est intéressant qu'un outil fournisse des mécanismes d'explication.

Certains, comme EMYCIN, ont un mécanisme d'explication complet construit dans l'outil lui-même, de sorte que tout système expert écrit dans ce langage peut directement accéder à ce mécanisme. Les outils qui ne comportent pas de tels mécanismes forcent l'ingénieur de connaissances à en concevoir un lorsqu'il construit le système expert.

Le mécanisme d'explication le plus courant utilise le raisonnement rétrospectif. Il explique comment le système a atteint un état particulier. Par exemple, l'utilisateur peut souhaiter savoir comment le système est arrivé à une certaine conclusion ou pourquoi le système a besoin de demander une certaine information. Le mécanisme d'explication permet au système de présenter la séquence des règles qui lui a permis d'atteindre la conclusion ou de décrire la règle qui l'a conduit à poser la question. Ces possibilités peuvent être offertes à l'utilisateur en mettant à sa disposition des primitives "WHY" et "HOW".

Les mécanismes d'explication peuvent aussi manipuler le raisonnement par hypothèse ("hypothetical reasoning") dans lequel le système explique ce qui se serait passé si un fait particulier avait été différent, et le raisonnement par défaut ("counterfact reasoning") dans lequel le système explique pourquoi une conclusion attendue n'a pas été atteinte.

5.4.4. Les éditeurs de bases de connaissances

Beaucoup d'outils de développement de systèmes experts comportent un éditeur de base de connaissances. Dans les cas les plus simples, il s'agit d'un éditeur de texte. Cependant, certains outils incorporent d'autres possibilités. Par exemple, EMYCIN utilise un aide-mémoire automatique ("automatic bookkeeping"). L'éditeur d'EMYCIN contrôle les changements réalisés par un utilisateur et enregistre automatiquement l'information pertinente sur l'évènement. Si l'utilisateur ajoute ou modifie une règle, l'éditeur mémorise automatiquement la donnée modifiée et le nom de l'utilisateur avec la règle. L'aide-mémoire est particulièrement utile lorsque plusieurs personnes modifient ou raffinent la base de connaissances.

Les éditeurs de bases de connaissances peuvent également assurer la vérification syntaxique. L'éditeur analyse la structure grammaticale des règles en fonction de la grammaire du langage défini pour écrire les règles du système expert. Lorsque l'utilisateur entre une règle syntaxiquement incorrecte, le système lui explique son erreur. Corriger de telles erreurs durant l'édition du programme plutôt que pendant l'étape de test permet de développer plus rapidement le système expert.

Une option particulièrement utile mais rarement disponible est la vérification de cohérence, dans laquelle le système contrôle la sémantique des règles et des données entrées. Il s'agit de vérifier si elles ne sont pas en contradiction avec des connaissances appartenant à la base de connaissances. L'éditeur réalise des vérifications sur les antécédents et les conséquents des règles introduites. Une contradiction survient, notamment, lorsque deux règles ayant les mêmes antécédents ont des conséquents différents ou lorsque l'antécédent d'une règle est un sous-ensemble d'une autre règle et qu'elles ont les mêmes conséquents. Lorsque survient un conflit, le système explique à l'utilisateur ce qui l'a causé et décrit les moyens de le résoudre.

Un éditeur de bases de connaissances performant comporte un mécanisme d'extraction de connaissances. Le système aide l'utilisateur à entrer de nouvelles connaissances dans le système. Il combine la vérification syntaxique et sémantique avec des explications sophistiquées pour permettre à des utilisateurs "naïfs" d'ajouter et de modifier des règles. Une telle facilité réduit le temps de développement du système expert et le temps de familiarisation des nouveaux utilisateurs du système. TEIRESIAS, par exemple, utilise un mécanisme d'extraction de connaissances.

Facultés Universitaires Notre-Dame de la Paix
Institut d'Informatique

DEVELOPPEMENT D'UN PROTOTYPE DE SYSTEME EXPERT POUR LA PLANIFICATION D'EXPERIENCES EN BIOCHIMIE

Y. LEBRUN et B. LHERMITTE

Mémoire présenté en vue de l'obtention du titre de licencié et maître en Informatique

Promoteur : M. NOIRHOMME-FRAITURE
Directeur : J. BARRETO

Septembre 1988

PARTIE 3

CONCEPTION DU SYSTEME

CHAPITRE 6. IDENTIFICATION

Lorsque nous avons commencé la conception du prototype d'un système expert d'aide à la purification de protéines, nous avons d'abord déterminé les aspects importants du problème. Nous nous sommes attachés à décrire le problème, à identifier les tâches que doit prendre en charge le système, à dégager les différents concepts du problème, à déterminer les objectifs de notre système.

Dans ce chapitre, nous présentons ces différents aspects. Leur identification est une première étape dans le développement du système expert. Certains éléments de ce chapitre ont déjà été abordés lors du chapitre 2. Nous avons tenu à les reprendre pour les identifier plus précisément et les placer dans notre démarche globale de conception.

6.1. IDENTIFICATION DU PROBLEME

6.1.1. Définition du problème

Nous nous intéressons à la purification des substances d'origine biologique. Par la suite, nous employons souvent le terme protéine au lieu de substance d'origine biologique.

Le problème se pose à une personne qui veut utiliser une protéine pour un travail quelconque. Elle ne dispose pas de cette protéine à l'état pur, mais uniquement dans un échantillon contenant également un grand nombre d'autres substances. Elle ne peut utiliser l'échantillon pour son travail car certaines substances nuiraient à son bon déroulement. Elle doit donc séparer la protéine qui

l'intéresse des contaminants. Pour réaliser correctement la purification, elle doit utiliser un plan de purification efficace. Nous appelons plan de purification, une suite de techniques de purification qui permet d'isoler une substance particulière contenue dans un échantillon complexe.

En fonction de l'usage que la personne souhaite faire de la protéine, elle peut classer les substances de l'échantillon en trois ensembles :

- la protéine à purifier
- les substances dont la présence empêche l'opérateur de faire usage de la protéine dans l'état actuel de l'échantillon. La purification doit permettre de les éliminer complètement
- les substances qui contaminent la protéine mais ne nuisent pas à son utilisation. Ces substances ne doivent pas nécessairement être éliminées.

En outre, la personne dispose d'informations propres à l'échantillon contenant le mélange : le volume et la quantité de substances biochimiques ou la concentration de l'échantillon.

6.1.2. Expertise nécessaire à la résolution du problème

Le problème n'est pas aisé à résoudre. Sa résolution requiert une certaine expertise. Les connaissances de base sont nombreuses. Il existe un nombre important de techniques de purification. Pour chacune d'elles, la pratique permet d'acquérir une expertise plus riche. Les mélanges d'origine biologique sont nombreux et variés.

6.1.2.1. Une connaissance approfondie des techniques de purification

Un plan de purification fait appel à plusieurs techniques de purification. Il en existe un grand nombre. Afin de circonscrire le problème, nous avons décidé, dans une première approche, de limiter la base de connaissances du système à trois techniques de purification : la chromatographie sur tamis moléculaire, la chromatographie sur échangeur d'ions et la précipitation.

Pour utiliser correctement une technique, une connaissance théorique est d'abord nécessaire. Un exposé théorique explique comment se réalise la séparation des substances, quel est le matériel à utiliser, comment le matériel doit être utilisé. Un exposé de ce type s'appuie sur la purification de substances connues, autrement dit sur des "cas d'école".

La pratique d'une technique de purification est indispensable pour en acquérir une connaissance approfondie. Certains critères, découverts par la pratique, permettent de prévoir si l'application d'une technique à un mélange peut entraîner une purification efficace. Certaines techniques sont inadéquates pour un mélange particulier. La pratique permet de tester réellement la réaction d'un mélange lors d'une purification.

6.1.2.2. Une connaissance approfondie de l'échantillon à purifier

Plus un biochimiste connaît les substances qui composent l'échantillon, mieux il peut utiliser les différentes techniques de purification qui sont à sa disposition. Une connaissance approfondie de l'échantillon permet à l'expert d'être plus efficace dans la construction d'un plan de purification.

Cependant, comme nous l'avons déjà signalé, un expert - a fortiori un biochimiste non-spécialiste - ne dispose bien souvent que de connaissances imparfaites sur un échantillon, à moins qu'il ne s'agisse d'un échantillon très simple. Une ou plusieurs substances d'un échantillon peuvent être inconnues de l'expert. Sa connaissance des caractéristiques d'une ou plusieurs substances peut être incomplète.

6.1.2.3. Une approche globale du problème

Pour construire des plans de purification efficaces, Il faut joindre, à la connaissance des techniques de purification et des mélanges biochimiques, une capacité de choisir et d'ordonnancer les techniques au sein d'un plan pour réaliser la purification la meilleure, la plus rapide, entraînant un minimum de pertes. L'expertise se mesure à la capacité de pouvoir agencer toutes les connaissances.

Les techniques sont en effet complémentaires. Construire un plan, c'est trouver un agencement complémentaire efficace des techniques.

Certaines techniques sont plutôt appliquées au début d'un plan pour purifier grossièrement l'échantillon. D'autres, en fin de plan, raffinent davantage la purification.

6.1.3. Prise en charge du problème par le système

Le système doit être capable d'aider l'utilisateur à construire un plan de purification efficace pour un échantillon quelconque. Il conseille l'utilisateur sur la suite des techniques à appliquer à un échantillon préalablement décrit, pour purifier cet échantillon et séparer la substance d'origine biologique à purifier des substances qui la contaminent.

Le plan doit être construit en constante interaction avec l'utilisateur. Celui-ci reçoit le conseil du système et prend une décision. C'est toujours en fonction de la décision de l'utilisateur que le système construit un plan.

Dans une première approche du problème, nous avons restreint les objectifs du système. Le système ne prend en charge que trois techniques de purification. L'explication du raisonnement du système n'est pas prévue. Les interfaces avec l'utilisateur et l'acquisition de connaissances ne sont qu'imparfaitement abordées. Il nous semble que notre système peut prendre la forme d'un "prototype-noyau". Certaines couches spécifiques, comme une interface ou un mécanisme d'explication, peuvent venir s'ajouter au prototype pour l'enrichir.

6.1.4. Tâches de la résolution du problème

La résolution du problème comporte trois tâches essentielles pour le système. Il doit les envisager dans un ordre précis :

- la détermination de l'échantillon à purifier
- la construction du plan de purification
- la présentation du plan construit à l'utilisateur.

Lorsqu'il envisage la construction d'un plan de purification, le système doit tenir compte d'un maximum d'informations sur les substances de l'échantillon. Dès lors il réunit les connaissances dont il dispose dans une description de l'échantillon à purifier. Une telle description constitue le point de départ pour la construction du plan.

Puisque le système ne dispose pas en général d'une description complète, il construit le plan dans un univers incomplet. Plus il dispose d'informations, et plus ces informations sont précises, meilleure sera sa construction.

La construction du plan de purification est le choix d'une suite adéquate de techniques à appliquer à un échantillon et l'évaluation du résultat de l'application de ces techniques pour construire la description de l'échantillon recueilli après leur application.

La présentation du plan de purification construit est la présentation de la solution offerte par le système.

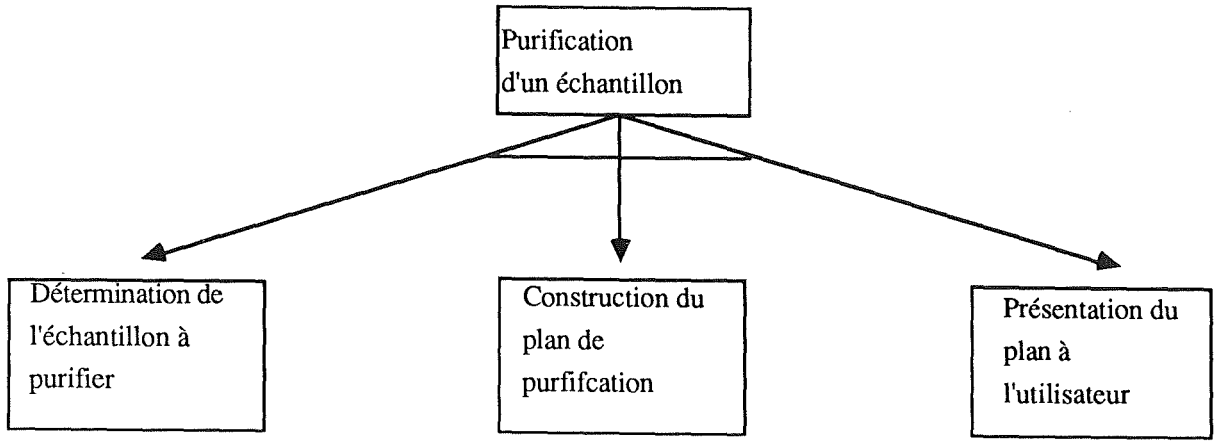

Figure 6.1.

6.1.5. Eléments d'une solution au problème

Une solution est constituée du plan construit et des résultats qu'il permet d'atteindre.

Le plan construit comporte différentes étapes. Pour chacune, la solution présente les conditions d'application de la technique (matériel de laboratoire, conditions expérimentales, etc.) et les résultats qui peuvent être obtenus en appliquant la technique (temps, perte en protéine, purification atteinte).

Les résultats globaux du plan sont le temps global, la perte globale de la protéine, la purification réalisée.

La solution doit permettre à l'opérateur non expérimenté d'appliquer chacune des techniques dans la pratique.

6.1.6. Obstacles qui empêchent la résolution du problème

Les mélanges d'origine biologique sont souvent peu connus, même des experts. En tout cas, les connaissances sont très réparties entre les biochimistes. Les recherches dans le domaine sont très longues et très difficiles. Aussi, le biochimiste ne base pas toujours une construction d'un plan de purification sur des connaissances précises. Il adopte parfois un plan de travail "par essais et erreurs" sur de petits échantillons du mélange. Certaines séparations efficaces peuvent ainsi être mises au point sans que l'on puisse apporter une explication claire des raisons du comportement des composants du mélange, sans qu'on connaisse véritablement les substances éliminées.

L'ingrédient essentiel dans un système expert est la connaissance du domaine. Il est indispensable que notre prototype puisse appuyer son raisonnement sur des connaissances aussi vastes que possible. La connaissance des mélanges biochimiques est une partie importante des connaissances dont doit disposer le système.

6.1.7. Incidence des obstacles sur le comportement du système

La qualité du conseil donné par le système dépend de la qualité et de la quantité des connaissances dont il dispose pour construire son conseil. Puisque ces connaissances sont rares, la mise au point d'un système ayant une efficacité réelle risque d'être longue. La mise au point doit

notamment permettre d'alimenter la base de connaissances du système. L'utilisateur peut apporter des connaissances inconnues du système. Pour la période de mise au point, on peut prévoir que le système suggère des expériences analytiques précises à l'expérimentateur afin d'acquérir des connaissances précises.

De toute manière, le système doit être capable de fournir un conseil même s'il ne dispose pas de beaucoup de connaissances. S'il consulte l'état des connaissances du système, l'utilisateur peut estimer la confiance qu'il accorde au conseil du système. Il peut aussi guider la construction du plan. Le système ne lui impose pas l'application d'une technique, ni les moyens à mettre en oeuvre pour l'appliquer. Toutes ces dispositions ajoutent de la souplesse qui doit amoindrir l'incidence des obstacles sur le système.

6.2. TYPE DE SYSTEME EXPERT CORRESPONDANT A LA SOLUTION

6.2.1. Spécificités d'un système expert de planification

Un plan est une suite d'actions à exécuter pour atteindre un objectif. Un système expert de planification est un programme qui, à partir d'un état initial et d'un état final, conçoit un plan dont l'exécution permet de passer de l'état initial à l'état final.

La première caractéristique importante d'un système expert de planification est l'état initial de l'objet. C'est sur cet objet que porte le problème sur lequel le système va travailler. Son état initial est décrit par l'utilisateur du système. Cette description n'est pas nécessairement exempte de lacunes ou d'imprécisions. Elle

se compose d'un certain nombre de caractéristiques prédéfinies par le concepteur du système.

La deuxième caractéristique est l'état final de l'objet. Il est assimilable à un état optimal, au sens où il s'agit de l'état souhaité par l'utilisateur, ou du moins, le plus proche de cet état en fonction des connaissances du système. L'état final est l'état dans lequel devra se trouver l'objet après que la suite d'actions conseillée par le système ait été exécutée. Il est également décrit par un ensemble de caractéristiques prédéfinies par l'auteur du système.

Une autre caractéristique d'un système de planification est le plan conseillé par le système. Il est inféré à partir de l'état initial de l'objet et de son état final. Le système utilise pour ce faire des connaissances enmagasinées dans une base de connaissances. Ces connaissances ont été fournies par un ou plusieurs experts dans le domaine spécifique du système.

La dernière caractéristique importante d'un système de planification est la manière interactive et incrémentale dont il travaille le plus souvent. Le système formule, raffine et tente de résoudre un problème décrit par l'état de l'objet via une conversation avec l'utilisateur. Il y a donc interactivité. Au cours de cette conversation, le système propose à l'utilisateur des solutions partielles possibles en fonction de la formulation courante du problème. Il conçoit son plan de manière incrémentale.

Il est à noter que cette dernière caractéristique n'est pas l'apanage des systèmes de planification. On la retrouve, par exemple, dans les systèmes de "design".

6.2.2. Classification de la solution envisagée

Le problème posé et la solution envisagée nous permettent de classer notre système parmi les systèmes de planification.

Si nous reprenons les termes de la définition d'un système expert de planification que nous avons donnée dans le point précédent, notre solution peut àtre reformulée de la manière suivante : il s'agit d'un programme qui, à partir de l'état initial d'un échantillon de substances biochimiques et de l'état final souhaité pour cet échantillon, conçoit un plan de purification dont l'exécution permet de faire passer l'échantillon de l'état initial à l'état final.

Pour ce qui est de l'état initial de l'échantillon, il se compose d'un ensemble de caractéristiques prédéfinies par le concepteur du système. Il décrit l'état de l'échantillon avant toute purification. Certaines caractéristiques sont connues du système, les autres sont fournies par l'utilisateur. Elles peuvent ne pas àtre complètes. Il est en effet assez rare que l'utilisateur connaisse toutes les protéines qui composent son échantillon. Pour les protéines connues, il peut ne pas connaître toutes les caractéristiques de ces protéines.

L'état final se compose d'un ensemble de caractéristiques définies par le concepteur du système. Ces caractéristiques sont semblables à celles décrivant l'état initial. Elles décrivent l'état de l'échantillon après purification. L'état final n'est pas spécifié de manière explicite par l'utilisateur. Le système considère qu'il a atteint l'état final lorsque l'échantillon est complètement purifié, c'est-à-dire quand l'utilisateur lui signale que la

purification obtenue lui semble suffisante, ou, lorsqu'en fonction des connaissances dont le système dispose, il ne lui est plus possible d'obtenir une meilleure purification.

Le plan conseillé par le système se compose d'une suite de techniques chromatographiques. Son application par l'utilisateur permettra de faire passer l'échantillon de l'état initial à l'état final. Le système dispose de connaissances concernant ces techniques chromatographiques qui lui ont été fournies par un ou plusieurs experts en biochimie. A partir de ces connaissances, de l'état initial de l'échantillon et de son état final, il construit un plan de purification.

La construction de ce plan se fait de manière interactive et incrémentale. Partant de l'état initial, le système propose à l'utilisateur l'application d'une technique. Il lui indique l'état de l'échantillon après application de cette technique. Il lui donne quelques renseignements concernant notamment le temps, le taux de purification et le matériel nécessaire à l'application de la technique. A partir de cela, l'utilisateur refuse ou accepte l'application de la technique. S'il refuse, le système lui en propose une autre sur base de l'état initial. S'il accepte, le système propose, de la même manière, la technique suivante à partir de l'état de l'échantillon résultant de l'application de la technique précédente. Ce processus se poursuit jusqu'à l'état final.

6.3. OBJECTIFS DU SYSTEME

La réalisation de la tâche par un outil de type système expert peut constituer un apport pour le laboratoire. D'abord, l'acquisition des connaissances auprès des experts

est, pour eux, une épreuve intéressante. Ils doivent structurer leur connaissance et formaliser un ensemble de pratiques informelles. Ensuite, le système doit permettre une distribution de l'expertise auprès de non-spécialistes. Nous pensons plus particulièrement aux opérateurs et aux étudiants du laboratoire. Le système peut apporter une aide au niveau de la réflexion sur l'enchaînement des techniques de purification au sein d'un plan de purification.

6.4. IDENTIFICATION DU ROLE DES PARTICIPANTS

Le laboratoire de biochimie cellulaire où nous avons travaillé est équipé en matériels informatiques. Le personnel y recourt largement dans le cadre de son travail. Cette connaissance de l'outil informatique en général a certainement facilité le contact. Nous avons fourni une information générale sur les systèmes experts aux personnes avec lesquelles nous avons plus spécialement travaillé. L'objectif de cette information était de leur faire percevoir le type de problème que nous souhaitions traiter et le type d'outil dont nous envisagions l'étude. Les entrevues que nous avons eues avec ces personnes ont essentiellement porté sur le domaine. Les problèmes informatiques auquels nous avons dû faire face ne leur ont jamais été exposés.

CHAPITRE 7. CONCEPTUALISATION

Ce chapitre décrit les aspects importants du problème que nous avons définis lors de l'étape d'identification. Nous développons les concepts et les relations du problème. Nous décrivons les différentes tâches que le prototype doit prendre en charge. Nous présentons également les connaissances acquises auprès des biochimistes concernant les trois techniques de purification retenues pour notre approche du problème. Ces connaissances ne sont pas complètes. Elles doivent être testées grâce au prototype, pour ensuite être affinées.

7.1. CONCEPTS ET RELATIONS

7.1.1. Les substances

Les substances biochimiques composent les échantillons sur lesquels le système travaille. Elles peuvent être d'origine biologique comme les protéines ou d'origine chimique comme les sels. Les substances à purifier sont toujours d'origine biologique.

Les caractéristiques des substances permettent de déterminer quelles techniques de purification peuvent les séparer. Ces caractéristiques sont :

- le nom
- la nature
- la masse moléculaire
- le point isoélectrique
- le domaine de stabilité
- le(s) domaine(s) de précipitation
- la stabilité dans le temps

Le nom de la substance est l'identification sous laquelle elle est connue des biochimistes : hémoglobine par exemple. La nature de la substance spécifie son origine. Sa valeur peut être protéine, enzyme, sel, etc. Les autres caractéristiques sont nécessaires au système pour envisager les trois techniques de purification retenues dans notre première approche du problème. Elles seront expliquées lorsque nous examinerons les conditions d'application des techniques en [7.3.]. Si d'autres techniques sont ajoutées à l'avenir, d'autres caractéristiques compléteront le concept de substance.

7.1.2. Les mélanges

Nous avons adopté le terme mélange pour désigner tout extrait de tissus cellulaires susceptible d'être manipulé dans une purification.

Un mélange est un ensemble de substances. Les caractéristiques d'un mélange sont :

- le nom
- l'identification des substances :
  - le nom de la substance
  - la proportion dans la quantité totale

7.1.3. Les échantillons

Un échantillon d'un mélange constitue l'élément manipulé lors de la purification. Il s'agit d'une occurrence du mélange. Ses caractéristiques sont :

- le nom du mélange
- le volume
- la concentration
- la quantité totale de substances

- la dissolution
- la présence de sels
- l'identification de la protéine à purifier. Elle se fait par le nom de la protéine
- l'identification des substances contaminantes à éliminer complètement. Elle se fait également par le nom des substances.

La dissolution exprime si l'échantillon est dissout dans un solvant. La présence de sels exprime si l'échantillon contient des sels. Ces deux caractéristiques sont importantes parce qu'elles déterminent s'il est nécessaire de réaliser des manipulations de préparation de l'échantillon avant l'application d'une technique de purification.

Il faut bien distinguer les concepts de mélange et d'échantillon. Le mélange est un extrait de tissus cellulaires. Il existe indépendamment de la purification. L'échantillon, par contre, est soumis à la purification. Il est constitué d'un mélange.

7.1.4. Les techniques

Les techniques de purification sont appliquées aux échantillons pour obtenir une séparation.

7.1.5. La relation "s'applique à"

La purification est une relation entre une technique de purification et un échantillon : la technique s'applique à l'échantillon. Cette relation est définie si et seulement si les conditions d'application d'une technique sont compatibles avec les caractéristiques de l'échantillon. Les

conditions d'application vont être précisées par la suite. La réalisation de la relation entre les deux concepts fournit un nouvel échantillon.

7.1.6. Les autres relations

Deux autres relations, moins essentielles, existent.

Le mélange contient des substances. Un mélange contient, en général, un grand nombre de substances. Une substance peut appartenir à plusieurs mélanges.

Un échantillon est une occurrence d'un mélange. L'échantillon possède toutes les caractéristiques du mélange dont il est une occurrence. Si, à l'avenir, des caractéristiques sont ajoutées pour préciser le concept de mélange, un échantillon, occurrence d'un mélange, possèdera également ces caractéristiques.

7.2. LES TACHES

Lors de l'identification du problème, nous avons esquissé un arbre-ET des trois tâches essentielles de la résolution du problème. Nous détaillons maintenant ces tâches pour étoffer l'arbre dégagé précédemment.

7.2.1. La détermination de l'échantillon à purifier

La première tâche que le système poursuit est une tâche de détermination de l'échantillon à purifier. L'objectif de cette tâche est de réunir un maximum d'informations permettant de caractériser l'échantillon.

Le système dispose de deux sources de connaissance pour déterminer l'échantillon : l'utilisateur qui lui soumet l'échantillon et ses propres connaissances accumulées lors de travaux précédents. Nous souhaitons que l'utilisateur ait à fournir un minimum de connaissances au système. Aussi, le système doit consulter en priorité ses connaissances, ne recourant à l'utilisateur que pour compléter son acquis.

Cette tâche se partage en plusieurs sous-tâches qui forment un sous-arbre-ET : l'identification du mélange qui constitue l'échantillon, la recherche des connaissances dont dispose déjà le système, l'interrogation de l'utilisateur.

Pour identifier le mélange, le système soumet à l'utilisateur un catalogue des mélanges qu'il connaît. Ce catalogue est composé des désignations de mélange. Le système suggère à l'utilisateur de désigner le mélange dans ce catalogue ou de lui soumettre un mélange inconnu.

Au cours des différents travaux qui lui seront soumis, le système acquerra des connaissances sur les mélanges biochimiques et les substances qu'ils contiennent. Une fois acquises, ces connaissances pourront être réutilisées par le système. La recherche de connaissances concerne donc le mélange dont l'échantillon est une occurrence et les différentes substances que contient le mélange. Le système présente à l'utilisateur un état de ses connaissances : les

différentes substances connues, ainsi que leurs caractéristiques connues.

L'interrogation de l'utilisateur a pour objectif d'enrichir l'acquis du système. L'utilisateur peut corriger une connaissance incorrecte ou introduire une nouvelle connaissance, sur les substances composant un mélange ou sur les caractéristiques d'une substance.

Si le système ne connaît pas le mélange, il signale son ignorance. L'utilisateur doit se résoudre à donner au système toutes les connaissances qu'il détient. Il cite les substances que contient le mélange et la proportion que chacune d'elles représente dans la quantité de matériel. Certaines de ces protéines peuvent déjà être connues du système. Il en présente alors les caractéristiques à l'utilisateur. Celui-ci a la possibilité de corriger ou de compléter. Pour les substances inconnues du système, l'utilisateur fournit une description la plus précise possible de leurs caractéristiques.

L'utilisateur doit compléter les caractéristiques de l'échantillon en donnant son volume, la quantité totale de matériel, une identification de la protéine à purifier et des substances à éliminer.

Le résultat de ces trois sous-tâches est une description aussi précise que possible de l'échantillon de départ. Cette description caractérise l'échantillon auquel le système doit appliquer la première technique.

7.2.2. La construction d'un plan de purification

La construction d'un plan de purification constitue la deuxième tâche de l'arbre-ET. Cette tâche comprend le choix de la technique la plus adéquate à appliquer à un échantillon et l'évaluation du résultat de l'application d'une technique pour construire la description de l'échantillon recueilli. Ces deux tâches doivent être appliquées de manière itérative jusqu'à l'obtention d'une purification satisfaisante. Le choix de la technique et l'évaluation de l'application de la technique forment un sous-arbre-ET.

Partant de la description d'un échantillon, le processus itératif réalise un choix de la technique de purification la plus adéquate à appliquer à cet échantillon. Lorsque le choix est réalisé, le système évalue l'application de la technique et construit la description de l'échantillon recueilli. C'est à partir de cette nouvelle description que la suite du plan peut être construite.

Le choix de la technique la plus adéquate à appliquer à un échantillon est le parcours d'un arbre-OU dans lequel chaque branche correspond à des critères de choix d'une technique de purification. Il s'agit de vérifier si les critères de choix d'une technique de purification sont compatibles avec les caractéristiques de l'échantillon à purifier et si les caractéristiques de l'échantillon sont telles que l'application permettrait de progresser dans la purification. Ces critères de choix sont présentées en 7.3. Dans cet arbre-OU, il ne faut pas envisager les techniques dans n'importe quel ordre. L'application de certaines techniques au début d'un plan est plus avantageuse que d'autres. L'inverse est vrai aussi. Par exemple, la précipitation isoélectrique est plutôt une technique appliquée au début d'un plan.

L'évaluation de l'application de la technique est également le parcours d'un sous-arbre-OU dans lequel chaque branche correspond à des critères d'évaluation de l'application d'une technique de purification. Elle consiste à déterminer le temps nécessaire à la réalisation de la technique, les pertes qu'elle occasionnera et la purification qu'elle permettra d'obtenir. De plus, il faut construire la description du nouvel échantillon à purifier. Enfin, le système présente un bilan de l'application de la technique à l'utilisateur : le matériel de laboratoire nécessaire à cette application, le temps et la perte estimés depuis le début du plan, la purification obtenue grâce à cette technique et depuis le début du plan, l'échantillon recueilli. C'est sur base de telles informations que l'utilisateur peut signifier au système de continuer le travail ou d'arrêter la construction, la purification obtenue le satisfaisant.

Lors de ces deux tâches de choix de la technique et d'évaluation de l'application de la technique, les conditions de l'application de la technique sont envisagées, comme le matériel de laboratoire à utiliser par exemple.

Si l'utilisateur a la possibilité d'arrêter la construction du plan de purification, le système peut parfois signifier lui-même l'arrêt de la construction. C'est le cas lorsque les conditions d'application de plus aucune technique ne sont réunies.

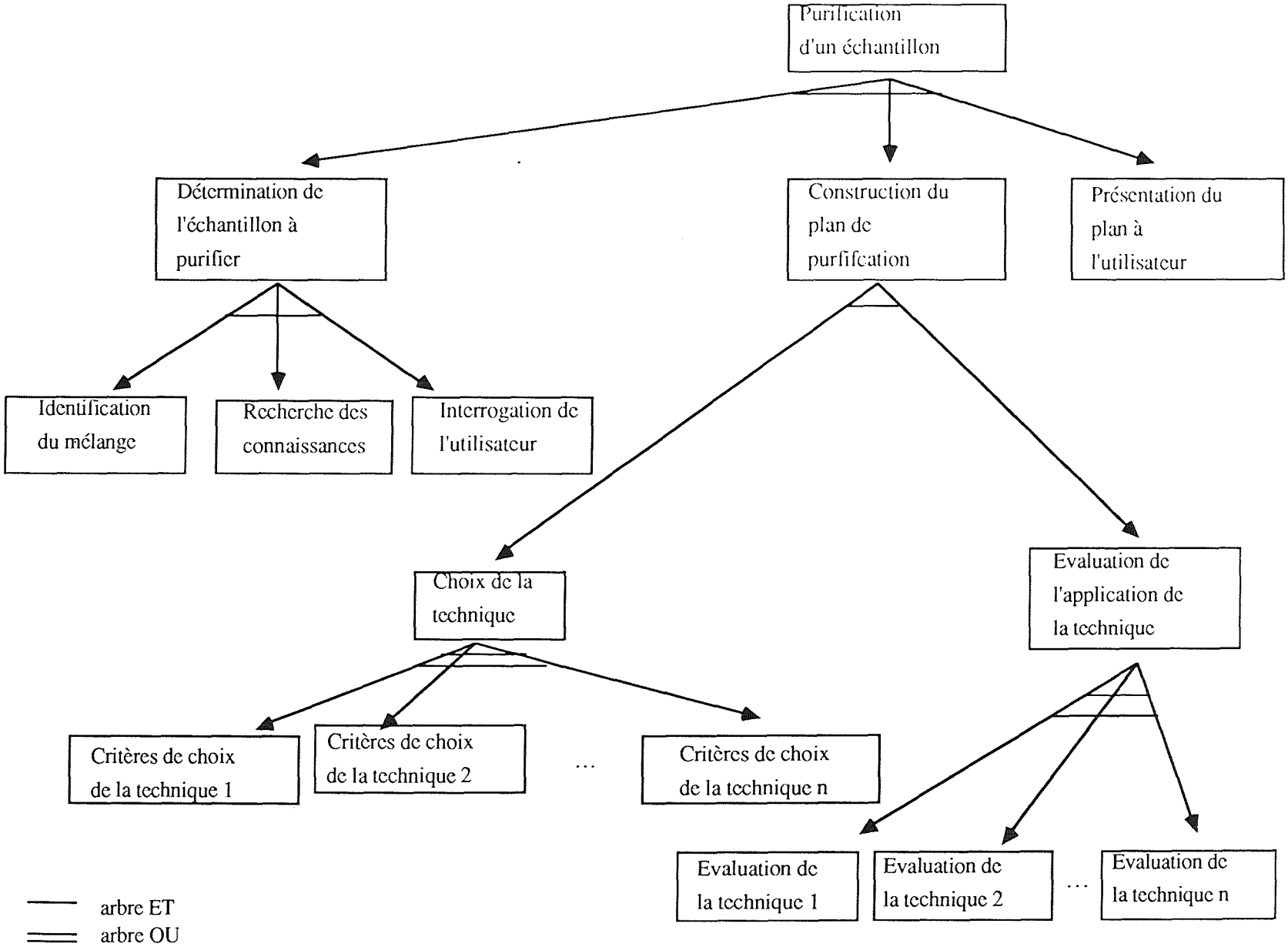
Purification d'un échantillon
Détermination de l'échantillon à purifier
Construction du plan de purification
Présentation du plan à l'utilisateur
Identification du mélange
Recherche des connaissances
Interrogation de l'utilisateur
Choix de la technique
Evaluation de l'application de la technique
Critères de choix de la technique 1
Critères de choix de la technique 2
...
Critères de choix de la technique n
Evaluation de la technique 1
Evaluation de la technique 2
...
Evaluation de la technique n
arbre ET
arbre OU

Figure 7.1.

7.3. LES CRITERES D'APPLICATION DES TECHNIQUES DE PURIFICATION

7.3.1. La chromatographie sur tamis moléculaire

7.3.1.1. Supports d'une chromatographie sur tamis moléculaire

Pour réaliser un tamis moléculaire, l'opérateur de laboratoire doit utiliser notamment une colonne, un gel, un tampon.

Il doit faire le choix le plus adéquat de ces supports parmi les différentes possibilités qui lui sont offertes au laboratoire.

Le système doit également réaliser ces choix parmi les supports qui lui sont connus.

La colonne

Chacune des colonnes utilisées au laboratoire est adaptée à certaines purifications. Les caractéristiques qui permettront au système de réaliser un choix adéquat de colonne sont le diamètre, la hauteur, la vitesse d'écoulement, le volume d'échantillon à déposer au maximum sur la colonne.

La vitesse d'écoulement est mesurée par le débit spontané de la colonne lorsqu'on réalise une pression en eau sur celle-ci, correspondant au 1/4 de sa hauteur. On utilise une vitesse d'élution correspondant au 3/4 du débit spontané. La pression exercée sur la colonne ne peut être trop élevée sous peine de tasser le gel et de réduire d'autant le débit.

Chaque colonne accepte un volume d'échantillon maximum. C'est une donnée cruciale : pour obtenir une séparation convenable, l'opérateur dépose un volume inférieur au volume d'échantillon maximum de la colonne [2.4.2.6.]. En tout cas, il ne peut déposer que des échantillons dont le volume est à peine supérieur au volume d'échantillon maximum. Par exemple, les échantillons déposés sur une colonne de 4cm de diamètre et 80cm de hauteur ne peuvent dépasser 20ml. En pratique, au laboratoire, l'opérateur dépose jusqu'à 30ml lorsqu'aucune autre solution ne lui semble meilleure.

Quatre colonnes seront initialement disponibles dans le système. Leurs caractéristiques sont :

- diamètre : 0.5cm
  hauteur : 80cm
  vitesse : 2ml/h
  volume maximum : 0.25ml

- diamètre : 1cm
  hauteur : 80cm
  vitesse :
  volume maximum : 1ml

- diamètre : 2cm
  hauteur : 80cm
  vitesse : 15ml/h
  volume maximum : 3.5ml

- diamètre : 4cm
  hauteur : 80cm
  vitesse : 40ml/h
  volume maximum : 20ml

Le gel

Plusieurs types de gel sont utilisés au laboratoire. Chacun de ces gels a ses propres caractéristiques. Ils sont utilisés en fonction de la taille des molécules et des propriétés chimiques des substances du mélange à purifier.

Dans un premier temps, le système doit connaître les gels qui sont habituellement utilisés au laboratoire : le Sephadex G-75, l'Ultrogel ACA-44, le Sephacryl S200, le Sepharose 4B. Les caractéristiques à retenir pour chacun d'eux sont le type, la marque, le domaine de fractionnement, le(s) coefficient(s) de non-séparation.

Le domaine de fractionnement du gel est une zone de poids moléculaires dans laquelle le gel assure une séparation des protéines de poids moléculaires différents. En-dessous de la limite inférieure se situe le volume total. Au-dessus de la limite supérieure se situe le volume mort [2.4.2.2].

Un coefficient de non-séparation exprime une estimation de l'efficacité de la séparation réalisée au moyen d'un gel particulier. Il permet de déterminer un domaine de non-séparation, zone de poids moléculaires dans laquelle le tamis moléculaire ne peut éliminer que partiellement les protéines. Par exemple, si la protéine à purifier a un poids moléculaire de 40000 dalton et le coefficient de non-séparation est 1.5, nous pouvons estimer que le tamis moléculaire éliminera complètement les protéines dont le poids moléculaire est inférieur à 26000 dalton (40000 / 1.5) ou supérieur à 60000 dalton (40000 * 1.5). Le domaine de non-séparation est alors (26000,60000).

Les caractéristiques des quatre gels à retenir sont :

- type : Sephadex G-75
  marque : Pharmacia
  domaine de fractionnement : 6000-65000
  coefficient de non-séparation : 1.5

- type : Ultrogel ACA-44
  marque : LKB
  domaine de fractionnement : 10000-120000
  coefficient de non-séparation : 1.5

- type : Sephacryl S200
  marque : Pharmacia
  domaine de fractionnement : 15000-160000
  coefficient de non-séparation : 2

- type : Sepharose 4B
  marque : Pharmacia
  domaine de fractionnement : 100000-10000000
  coefficient de non-séparation : 3

Il faut noter que le coefficient de non-séparation donné ci-dessus pour chaque gel a été estimé pour une colonne d'une hauteur de 80cm, c'est-à-dire la hauteur des colonnes qui seront initialement disponibles pour le système. Si des colonnes de hauteur différente sont introduites, le coefficient sera différent et devra être mémorisé. C'est donc en fonction de la hauteur de la colonne utilisée que le coefficient de non-séparation peut être déterminé.

Pour chacun des gels décrits existent différentes tailles de grains : normale, fine, superfine. Plus le grain est fin, mieux la séparation se déroule, mais plus elle est lente. Pour les travaux préparatifs réalisés au

laboratoire, on utilise toujours un grain de taille fine. Nous considérons que les gels connus du système sont de taille fine.

Le tampon

Le tampon généralement utilisé est un tampon phosphate 0.01M à 4°C et à pH 7.4, contenant du NaCl 0.15M. Il permet de conserver un équilibre physiologique des substances. Parfois, on utilise un tampon TRIS-HCl 0.05M à pH 7.4 (TRIS = Tris (hydroxymethyl) aminométhane). Leur "pouvoir tampon" est caractérisé par leur pK [2.4.1.2.] :

- nom : tampon phosphate
  pK : 7.2

- nom : tampon TRIS
  pK : 8.2

7.3.1.2. Critères pour le choix de l'application de la chromatographie sur tamis moléculaire

Lorsque le système réalise un choix quant à la technique la plus adéquate à appliquer au mélange à purifier, il peut envisager la technique du tamis moléculaire. Les critères décrits ci-dessous doivent lui permettre d'estimer si l'application de la chromatographie sur tamis moléculaire est intéressante pour l'échantillon à purifier.

Pour cette évaluation, le système doit tenir compte de trois critères :

1. Le nombre de substances doit être peu important
2. Le volume d'échantillon doit être peu important

3. Les poids moléculaires doivent être suffisamment différents

7.3.1.2.1. Le nombre de substances doit être peu important

Lorsque le nombre de substances contenues dans le mélange est trop important, la séparation sur le tamis moléculaire est mauvaise. Les bandes de substances se chevauchent fortement. La protéine à purifier se disperse parce qu'elle ne représente qu'un faible pourcentage du mélange.

Cependant, se baser sur le nombre de substances n'est pas un critère convenable. En effet le système, comme l'opérateur de laboratoire, ne connaît généralement pas toutes les substances du mélange. Aussi, il est plus fiable de se baser sur la proportion que représente la protéine à purifier dans l'ensemble du mélange : la quantité de protéine à purifier présente dans le mélange doit être supérieure à 10% de la quantité totale de matériel.

Nous constatons que la technique du tamis moléculaire est plutôt appliquée après plusieurs étapes de purification, lorsque la protéine représente une proportion relativement importante du mélange.

7.3.1.2.2. Le volume d'échantillon doit être peu important

Le volume d'un mélange à déposer sur une colonne de tamis moléculaire est limité [2.4.2.6.]. Nous pouvons énoncer des règles fort générales à propos du volume :

1° plus le volume du mélange est petit, meilleures sont les conditions pour l'application d'un tamis moléculaire.

Dans ce cas, la colonne à préparer est plus petite et donc moins coûteuse. Le tamis moléculaire dure moins longtemps et entraîne moins de pertes.

2° la technique, telle qu'elle est utilisée au laboratoire, accepte des mélanges d'un volume maximum de 20ml.

C'est le volume maximum de la plus grosse colonne utilisée.

3° on peut déposer des mélanges d'un volume allant jusqu'à 30ml.

Si l'application d'un tamis moléculaire peut apporter une purification importante, mais que le volume est compris entre 20 et 30ml, on peut réaliser tout de même le tamis, en sachant que la résolution de la colonne sera moins bonne et les pertes seront plus importantes qu'avec un plus petit volume.

Le système doit s'assurer que le volume du mélange respecte les règles énoncées ci-dessus : le volume doit être le plus petit possible, de préférence inférieur à 20ml et, en tout cas, inférieur à 30ml.

Ceci peut amener le système à suggérer une concentration de l'échantillon sur une membrane d'ultra-filtration de manière à ramener le volume dans des limites acceptables. Cependant la concentration du matériel dans l'échantillon doit respecter également certaines règles pour pouvoir appliquer un tamis moléculaire [2.4.2.7.] :

1° la "concentration idéale" de matériel dans l'échantillon est de 10mg de matériel par ml de mélange

2° la technique peut être appliquée pour autant que la concentration soit comprise entre 2 et 10 mg par ml

3° une concentration de 4 à 5mg par ml est une bonne concentration

Le système doit veiller à respecter cette échelle de concentration. Il faut plutôt s'approcher de la "concentration idéale".

Avec l'aide de l'utilisateur, le système doit ajuster le volume et la concentration du mélange, déterminer s'il est adéquat de réaliser une concentration sur une membrane d'ultra-filtration. L'utilisateur ne doit jamais perdre de vue qu'une concentration sur membrane entraîne des pertes de matériel. Le système doit le lui rappeler.

Pour respecter ces règles, le système peut envisager plusieurs situations :

a) le volume de l'échantillon est supérieur à 20ml et la concentration réelle de l'échantillon est plus forte que la "concentration idéale". Le tamis moléculaire ne semble pas adéquat : l'échantillon ne peut en principe pas être concentré davantage. Le volume ne peut donc être réduit. L'utilisateur peut cependant décider de la suite du travail entre :

- réaliser un tamis moléculaire avec un volume trop important et une concentration qui n'est pas "idéale". Cette solution est envisageable si le volume n'est pas supérieur à 30ml. La concentration ne doit pas s'éloigner trop de la "concentration idéale".
- avant d'appliquer le tamis moléculaire, concentrer l'échantillon jusqu'à un volume inférieur ou à peine supérieur à 20ml en acceptant de travailler à une concentration plus élevée. La concentration ne doit pas s'éloigner trop de la "concentration idéale".

- séparer l'échantillon en plusieurs parties, diluer chacune de ces parties pour obtenir une concentration proche de la "concentration idéale" et un volume inférieur ou à peine supérieur à 20ml, et réaliser plusieurs tamis moléculaires. Cette solution risque d'être coûteuse en temps
- ne pas réaliser de tamis moléculaire

b) le volume de l'échantillon est supérieur à 20ml et la concentration réelle de l'échantillon est "idéale". Dans la mesure où le volume de l'échantillon dépasse à peine 20ml, le tamis moléculaire peut être utilisé. On peut également penser à concentrer l'échantillon pour diminuer le volume. L'utilisateur peut décider de la suite du travail entre :

- réaliser un tamis moléculaire avec un volume trop important. Cette solution est envisageable si le volume n'est pas supérieur à 30ml
- avant d'appliquer le tamis moléculaire, concentrer l'échantillon pour obtenir un volume à peine supérieur à 20ml et accepter de travailler à une concentration plus élevée. La concentration ne doit pas s'éloigner trop de la "concentration idéale"
- séparer l'échantillon en plusieurs parties d'un volume inférieur ou à peine supérieur à 20ml, et réaliser plusieurs tamis moléculaires. Cette solution risque d'être coûteuse en temps
- ne pas réaliser de tamis moléculaire

c) le volume de l'échantillon est supérieur à 20ml et la concentration réelle de l'échantillon est plus faible que la "concentration idéale". Le système peut proposer à l'utilisateur de réaliser une concentration de l'échantillon afin d'obtenir un volume inférieur ou égal à 20ml. L'utilisateur peut décider de la suite du travail entre :

- réaliser un tamis moléculaire avec un volume trop important et une faible concentration. Cette solution est envisageable si le volume n'est pas supérieur à 30ml et la concentration comprise entre 2 et 10mg par ml
- avant d'appliquer le tamis moléculaire, concentrer l'échantillon jusqu'à un volume à peine supérieur ou inférieur à 20ml en augmentant la concentration. Il faut faire en sorte que la concentration reste proche de la "concentration idéale"

d) le volume de l'échantillon est inférieur à 20ml et la concentration réelle de l'échantillon est plus forte que la "concentration idéale". Pour autant que la concentration ne soit pas trop supérieure à 10mg par ml, le tamis peut être appliqué. L'utilisateur peut décider de la suite du travail entre :

- appliquer le tamis moléculaire à l'échantillon. Cette solution est envisageable si la concentration est proche de 10mg par ml
- avant d'appliquer le tamis moléculaire, diluer l'échantillon pour améliorer la concentration. Le volume doit cependant rester proche de 20ml au maximum

e) le volume de l'échantillon est inférieur à 20ml et la concentration réelle de l'échantillon est "idéale". Le volume de l'échantillon est donc bien adapté et peu important. Le tamis moléculaire peut être appliqué

f) le volume de l'échantillon est inférieur à 20ml et la concentration réelle de l'échantillon est plus faible que la "concentration idéale". Le système peut proposer à l'utilisateur de réaliser une concentration de l'échantillon afin d'arriver à une "concentration idéale" et à un volume

plus petit. L'utilisateur peut décider de la suite du travail entre :

- appliquer le tamis moléculaire à l'échantillon
- avant d'appliquer le tamis moléculaire, concentrer l'échantillon. La concentration de l'échantillon doit rester comprise entre 2 et 10mg par ml, aussi proche que possible de 10mg par ml.

Lors de cette étape, le système décide donc, avec l'aide de l'utilisateur, si le tamis moléculaire peut être appliqué, si une concentration sur une membrane d'ultra-filtration est réalisée avant d'appliquer le tamis moléculaire, et évalue le volume qui est déposé sur la colonne.

7.3.1.2.1. Les poids moléculaires doivent être suffisamment différents

Les poids moléculaires des substances à séparer doivent être suffisamment différents afin d'obtenir une bonne séparation. Si deux substances ont des poids moléculaires proches, elles sortiront de la colonne ensemble et donc ne seront pas séparées par le tamis moléculaire.

Critère : on considère que les poids moléculaires sont assez différents si, parmi les substances à éliminer, le nombre de substances qui pourront être éliminées complètement est plus important que le nombre de substances qui resteront et contamineront le mélange recueilli.

Pour évaluer le nombre de substances que le tamis moléculaire permet d'éliminer complètement, il faut tenir compte de trois éléments :

- de la colonne utilisée
- du gel utilisé

- des poids moléculaires des différentes substances connues dans le mélange

Le volume du mélange à déposer sur la colonne est connu. Le système réalise le choix de la colonne : la plus petite colonne acceptant un échantillon d'un tel volume. Ce choix doit être soumis à l'utilisateur. Celui-ci ne dispose peut-être pas de cette colonne, ou encore, ne désire pas l'utiliser. Il doit avoir la possibilité d'imposer son choix au système, pour autant que ce choix soit cohérent. L'utilisateur doit également avoir la possibité de suggérer l'utilisation d'une nouvelle colonne, à condition d'en donner une description complète. Cette description doit comprendre toute les informations connues pour chaque colonne et les coefficients de non-séparation de cette nouvelle colonne pour tous les gels connus du système. Une description imcomplète implique soit l'impossibilité pour le système de travailler avec cette colonne, soit l'impossiblité d'utiliser un gel particulier avec cette colonne.

Ensuite, le système réalise le choix du gel. Il cherche quel est le gel qui peut permettre d'éliminer complètement le plus de substances. D'abord, il faut que le poids moléculaire de la protéine à purifier soit plus ou moins au centre du domaine de fractionnement du gel. Ainsi sont évités certains problèmes qui pourraient résulter du fait que le poids moléculaire de la protéine à purifier est proche des limites du domaine de fractionnement. Si le poids moléculaire de la protéine à purifier est proche de la limite supérieure du domaine de fractionnement, nous risquons d'obtenir à la chromatographie, un recouvrement entre le pic du volume mort et le pic de la protéine à purifier. Nous n'obtiendrons donc pas une bonne séparation avec les substances de poids moléculaire important qui constituent le volume mort. Pour un poids moléculaire de la

protéine à purifier proche de la limite inférieure, le problème se pose avec le volume total. Un premier critère quant au choix du gel est donc : le domaine de non-séparation de la protéine est inclus dans le domaine de fractionnement du gel. Parmi les gels ayant satisfait ce premier critère, le choix du gel doit se porter sur celui qui répond le mieux au second critère : éliminer complètement le plus grand nombre de substances.

Enfin, le système peut évaluer si le gel le plus adéquat sur la colonne la plus adéquate permet d'éliminer complètement plus de substances qu'il ne restera de contaminants. Il peut donc décider si les poids moléculaires sont suffisamment différents. Une réponse négative signifie que le système conseillera à l'utilisateur de ne pas appliquer le tamis moléculaire. Cependant, en dernier recours, celui-ci reste seul juge. Il peut estimer qu'une application d'un tamis moléculaire est utile même si le nombre de substances éliminées est moins importants que le nombre de contaminants conservés.

7.3.1.3. Evaluation de l'application de la chromatographie sur tamis moléculaire

Lorsque le système a décidé d'appliquer la technique du tamis moléculaire, il doit en évaluer le résultat. Il s'agit d'établir la description la plus complète possible de l'échantillon recueilli après l'application.

7.3.1.3.1. Choix du tampon

Le domaine de "pouvoir tampon" s'étend sur l'intervalle [pK-1,pK+1] [2.4.1.2.]. Le système doit conseiller l'utilisation d'un tampon dont le domaine de "pouvoir tampon" est compatible avec le domaine de stabilité de la

protéine à purifier. Il faut travailler à un pH, si possible proche de 7.4, assurant la stabilité de la protéine à purifier. Le pH est maintenu constant par le tampon.

7.3.1.3.2. Evaluation du volume de l'échantillon recueilli

Le volume probable de l'échantillon recueilli après l'application du tamis moléculaire dépend du volume de l'échantillon de départ et du volume de la colonne. En général, nous pouvons estimer que le volume obtenu sera égal à 5% du volume total de la colonne auquel il faut ajouter le volume de l'échantillon.

7.3.1.3.3. Evaluation de la nature et de la quantité des substances dans l'échantillon recueilli

Lorsque le système a décidé l'application du tamis moléculaire, il a fait le choix du gel et de la colonne. Dès lors, le coefficient de non-séparation est connu. Il permet d'estimer un domaine de non-séparation. Le tamis moléculaire permettra d'éliminer complètement toutes les substances dont le poids moléculaire n'est pas compris dans ce domaine de non-séparation. Les substances dont le poids moléculaire est compris dans ce domaine contamineront d'autant plus le résultat que leur poids moléculaire est proche du poids moléculaire de la protéine à purifier.

Afin d'estimer le pourcentage de contamination d'une substance dont le poids moléculaire est compris dans le domaine de non-séparation, nous considérons que si le poids moléculaire de la substance est

- égal à la limite inférieure ou supérieure du domaine, la contamination sera de 0%
- égal au poids moléculaire de la protéine à purifier, la contamination sera de 100%

- compris entre la limite inférieure du domaine et le poids moléculaire de la protéine à purifier, le pourcentage de contamination d'une substance est une fonction linéaire passant par les points (limite inférieure,0%) et (poids moléculaire de la protéine à purifier,100%) du poids moléculaire de cette substance.
- compris entre le poids moléculaire de la protéine à purifier et la limite supérieure du domaine, le pourcentage de contamination d'une substance est une fonction linéaire passant par les points (poids moléculaire de la protéine à purifier,100%) et (limite supérieure,0%) du poids moléculaire de cette substance.

Il faut remarquer que nous avons choisi une fonction linéaire en première approche. Ce n'est sans doute pas correct. A l'avenir, des observations complémentaires devront permettre de corriger ce défaut.

7.3.1.3.4. Evaluation de la durée du tamis moléculaire

La durée du tamis moléculaire dépend de la colonne utilisée pour le réaliser.

A partir des informations connues à propos de la colonne choisie, la durée du tamis moléculaire est estimée à partir de la formule :

durée = volume total de la colonne/ vitesse

La durée d'une chromatographie sur tamis moléculaire est de plusieurs heures, 15 à 20 heures en général.

7.3.2. La chromatographie sur échangeur d'ions

7.3.2.1. Supports d'une chromatographie sur échangeur d'ions

Pour réaliser un échangeur d'ions, l'opérateur de laboratoire doit utiliser, notamment, une colonne, un gel, un tampon.

Il doit faire le choix le plus adéquat de ces supports parmi les différentes possibilités qui lui sont offertes au laboratoire.

De même, le système doit réaliser ces choix parmi les supports qui lui sont connus.

La colonne

La colonne est caractérisée par sa hauteur et son diamètre. Le système ne doit retenir aucune caractéristique concernant les colonnes utilisées pour une chromatographie sur échangeur d'ions.

Le gel

Le système doit retenir le type, la marque et la charge électrique de chaque gel.

Dans un premier temps, le système ne retiendra que les gels suivants, qui sont habituellement utilisés au laboratoire :

type	: CM
marque	: Sephadex C50
charge électrique	: négative

type : SP
marque : Sephadex C50
charge électrique : négative

type : DEAE
marque : Sephadex A50
charge électrique : positive.

Les gels négatifs fixeront les protéines chargées positivement alors que les gels positifs fixeront les protéines chargées négativement.

Le tampon

Dans un premier temps, le système retiendra les tampons habituellement utilisés au laboratoire et leur pK :

nom : acide acétique
pk : 4.75

nom : phosphate
pk : 7.2

nom : citrate
pk : 3.1 / 4.7 / 5.4

nom : cacodilate
pk : 6.2

nom : glycine
pk : 9.9

nom : TRIS-HCl
pk : 8.2.

Le tampon citrate est très souvent utilisé au laboratoire.

7.3.2.2. Criteres pour le choix de l'application de la chromatographie sur échangeur d'ions

Le problème consiste à évaluer si l'application de la chromatographie sur échangeur d'ions est intéressante pour un mélange biologique.

Pour cette évaluation, le système doit tenir compte du critère suivant :

> Les points isoélectriques (PI) des protéines contaminantes doivent être suffisamment différents du PI de la protéine à purifier.

Les PI doivent être suffisamment différents

Les PI des protéines contenues dans le mélange doivent être suffisamment différents afin d'obtenir une bonne séparation. Si deux protéines ont des PI proches, elles sortiront de la colonne ensemble et donc ne seront pas séparées par l'échangeur.

Critère : les PI sont suffisamment différents si le nombre de protéines qui pourront être éliminées complètement est plus important que le nombre de protéines qui resteront et contamineront le résultat.

Pour séparer complètement des protéines sur échangeur d'ions, il faut que ces protéines aient des PI différents de 1 au moins.

Dès lors le nombre de protéines qui seront complètement éliminées sera égal au nombre de protéines dont le PI est différent du PI de la protéine à purifier de 1 ou plus (>=1). Le nombre de protéines qui contamineront la protéine à purifier sera égal au nombre de protéines dont le PI est différent du PI de cette protéine de moins de 1 (<1).

7.3.2.3. Evaluation de l'application de la chromatographie sur échangeur d'ions

Après avoir décidé de l'application d'un échangeur d'ions, le système doit conseiller l'utilisateur sur différents choix relevant de l'application de la technique elle-même et lui indiquer l'échantillon résultant de son application.

Choix du gel

Dans les chromatographies sur échangeur d'ions, on cherche souvent à obtenir une protéine dans un mélange. Pour ce faire, la stratégie appliquée est celle qui consiste à fixer la protéine étudiée sans fixer le reste du mélange (les contaminants de la protéine).

Le programme, connaissant le domaine de stabilité et le PI de la protéine à purifier, conseillera à l'utilisateur de travailler à un pH qui, tout en restant dans le domaine de stabilité, est le plus éloigné possible du PI. De cette manière, on obtiendra, pour la protéine, la charge la plus forte en fonction du domaine de stabilité et la protéine se fixera mieux sur la colonne. Ce pH prendra donc la valeur de la borne du domaine de stabilité la plus éloignée du PI.

Pour le choix du gel proprement dit, suivant que le pH est supérieur ou inférieur au PI de la protéine à purifier, le programme conseillera un gel positif ou négatif respectivement.

Dans le cas d'un gel négatif, deux choix sont encore possibles :

- si le pH est inférieur ou égal à 5, le système conseillera le SP, qui est plus dénaturant que le CM mais qui garde sa forte charge négative dans une zone de pH bas
- si le pH garde des valeurs supérieures à 5, le système conseillera plutôt le CM.

Choix du volume de gel

Pour les types de gel connus du système, la quantité maximale de protéine qu'il est possible de fixer est de 3.75 grammes de protéine par gramme de gel sec.

On garde, pour la chromatographie, une marge de sécurité de 3 par rapport à la quantité maximale de fixation.

Dès lors, le système conseillera le volume de gel sur base du rapport suivant :

3.75 grammes de protéines / 3 grammes de gel sec.

Un gramme de gel sec correspond habituellement à un volume de 18 ml de gel gonflé. Ce volume dépend essentiellement du type de gel et de la force ionique du tampon utilisé. On ne travaille pas, d'habitude, avec une force ionique inférieure à 0.1. En effet, sous ce seuil, il risque de se produire un gonflement exagéré du gel.

Choix de la colonne

La hauteur de la colonne utilisée varie entre 10 et 20 cm. Il n'existe pas de règle précise permettant de spécifier précisément la hauteur adéquate.

Le diamètre de la colonne est fonction du volume de l'échantillon et de la hauteur de la colonne. Le système conseillera l'utilisateur en appliquant la formule :

diamètre = 2 * sqrt(volume/hauteur*3.14)

Choix du tampon

Le système conseillera le tampon a utiliser de manière à ce que le pH auquel il a décidé de travailler et le PI de la protéine à purifier soient compris dans la zone tampon. Ainsi, lors du décrochage, l'utilisateur pourra modifier le pH jusqu'au PI tout en restant dans la zone tampon.

Si un tel tampon n'existe pas, le système conseillera, pour la même raison, celui qui comprend le pH et dont une des deux bornes de la zone tampon est la plus proche du PI de la protéine à purifier.

Il présentera à l'utilisateur le tampon et son pK. Si plusieurs tampons sont envisageables, il les proposera tous.

Choix du type d'élution

Pour décrocher les protéines, l'utilisateur a d'une part le choix entre modifier le PH ou modifier la force ionique μ.

Si le PI de la protéine à purifier se trouve en dehors du domaine de stabilité de cette protéine, le système

conseillera de modifier μ. En effet, modifier le pH consisterait à le rendre proche ou même égal au PI, ce qui nuirait à la stabilité de la protéine.

Si ce PI se trouve dans le domaine de stabilité, il n'existe pas de critère de choix entre modifier μ ou modifier pH. Le système conseillera alors de modifier l'un puis l'autre. De cette manière, on évite les effets négatifs rencontrés lorsqu'on pousse une des deux modifications à l'extrême. L'utilisateur modifiera le pH en veillant à ne pas sortir de la zone tampon.

La modification de la force ionique s'effectue habituellement à l'aide du NaCl. Cependant, il arrive qu'un sel plus fort soit utilisé : le NaSCl. Ce sel pose des problèmes d'élimination du fait de sa viscosité. Dans ce cas, on effectue une dialyse après l'échangeur pour éliminer la viscosité avant d'envisager la technique suivante.

D'autre part, l'utilisateur peut choisir de décrocher par gradient ou par modification des conditions en étapes. Le système ne dispose d'aucun critère qui lui permette de conseiller une manière plutôt qu'une autre. L'utilisateur est seul juge. Il devra cependant faire connaître son choix au système car celui-ci aura une influence sur l'efficacité de la purification et sur le volume d'échantillon obtenu à la sortie de la colonne.

Evaluation du volume de l'échantillon résultat

Le volume probable de l'échantillon après l'application de l'échangeur d'ions dépend du volume du gel et du type de décrochage.

En général, nous pouvons estimer que le volume obtenu sera égal à 50% du volume total de gel dans le cas où l'on

effectue le décrochage en une étape. Si le décrochage est effectué par gradient, le volume probable à la sortie sera égal au volume total de gel.

Evaluation de la nature et de la quantité de protéines dans le résultat

Un échangeur d'ions permettra d'éliminer complètement toutes les protéines dont le PI n'est pas compris dans l'intervalle [PI de la protéine à purifier - 1 , PI de la protéine à purifier + 1]. Les protéines dont le PI est compris dans cet intervalle contamineront d'autant plus le résultat que leur PI est proche du PI de la protéine à purifier.

Afin d'estimer le pourcentage de contamination d'une protéine dont le PI est compris dans l'intervalle, nous considérons que si le PI de la protéine est :
- égal à la limite inférieure ou supérieure de l'intervalle, la contamination sera de 0%
- égal au PI de la protéine à purifier, la contamination est de 100%
- compris entre la limite inférieure de l'intervalle et le PI de la protéine à purifier, le pourcentage de contamination d'une protéine est une fonction linéaire passant par les points (limite inférieure,0%) et (PI de la protéine à purifier,100%) du PI de cette protéine.
- compris entre le PI de la protéine à purifier et la limite supérieure de l'intervalle, le pourcentage de contamination d'une protéine est une fonction linéaire passant par les points (PI de la protéine à purifier,100%) et (limite supérieure,0%) du PI de cette protéine.

Il faut remarquer que nous avons choisi un système linéaire en première approche. Ce n'est sans doute pas

correct. A l'avenir, des observations complémentaires devront nous permettre de corriger ce défaut.

Evaluation de la durée de l'échangeur d'ions

La durée moyenne d'un échangeur d'ions est estimée à environ 4 heures. Ces 4 heures se répartissent entre 1 heure de fixation, 1 heure de lavage et 2 heures de gradient.

Cette durée peut éventuellement varier car on peut modifier le débit.

Evaluation de la perte d'activité de la protéine

La perte d'activité due à chaque manipulation est estimée à environ 10% pour un échangeur d'ions. Cela signifie qu'il faudra considérer une perte de 10% pour l'application de l'échangeur lui-même et, qu'il faudra éventuellement ajouter 10% pour chaque manipulation consécutive à l'échangeur et précédent les autres techniques (dialyse, par exemple).

A cette perte d'activité due à la manipulation, il faudra ajouter la perte par dénaturation de la protéine au cours du temps, pour obtenir la perte totale d'activité.

7.3.3. LA PRECIPITATION

7.3.3.1. Supports d'une précipitation

Pour réaliser une précipitation, l'opérateur doit disposer d'un récipient contenant le mélange à purifier et d'un agent de précipitation. On utilise habituellement une solution de sulfate ammonique saturée à 4°C et à pH 7.

7.3.3.2. Critère pour le choix de l'application de la précipitation

Quand la précipitation est possible, au début de la purification, il faut la réaliser car elle permet une purification partielle mais aisée à réaliser et rapide (1 heure). Il n'y a qu'un seul critère qui puisse limiter l'application d'une précipitation. Il faut que la protéine à purifier précipite complètement avant que le volume de l'agent de précipitation n'atteigne 60 % du volume total de l'échantillon (40 % de l'échantillon constitué de la solution de départ, 60 % constitué de la solution de l'agent de concentration.

Critère : La protéine à purifier doit précipiter avant 60 %.

7.3.3.3. Evaluation de l'application de la précipitation

Le système doit établir la description la plus complète possible de l'échantillon recueilli après l'application de la précipitation.

Choix du volume de l'agent de précipitation

Le système doit conseiller à l'utilisateur le volume de la solution contenant l'agent de précipitation à déposer progressivement dans le récipient. Ce volume est exprimé par un pourcentage. Il s'agit du pourcentage du volume de l'agent de précipitation auquel la protéine à purifier aura complètement précipité. Il correspond à la limite supérieure du domaine de précipitation de la protéine à purifier. Le domaine de précipitation d'une protéine est un intervalle dont la limite inférieure est le pourcentage du volume de l'agent de précipitation auquel la protéine commence à précipiter et la limite supérieure est le pourcentage du volume de ce même agent auquel la protéine a totalement précipité. L'utilisateur doit augmenter la concentration de l'agent de précipitation dans le mélange jusqu'à ce pourcentage.

Evaluation de la nature et de la quantité de protéines dans l'échantillon recueilli

Pour déduire la quantité et la nature des protéines dans l'échantillon après purification, le système se base sur le domaine de précipitation des protéines connues du mélange.

Afin de déduire le pourcentage de contamination d'une protéine, on compare le pourcentage de l'agent de précipitation auquel on a travaillé (limite supérieure du domaine de précipitation de la protéine à purifier) avec le domaine de précipitation de la protéine :

- si le pourcentage de l'agent est inférieur ou égal à la limite inférieure du domaine, la contamination sera de 100%

- si le pourcentage de l'agent est supérieur ou égal à la limite supérieure du domaine, la contamination sera de 0%
- si le pourcentage de l'agent est compris dans le domaine de précipitation de la protéine, le pourcentage de contamination est une fonction linéaire du pourcentage de l'agent passant par les points (limite inférieure, 0%) et (limite supérieure, 100%).

Il faut remarquer que nous avons choisi un système linéaire en première approche. Ce n'est sans doute pas correct. A l'avenir, des observations complémentaires devront nous permettre de corriger ce défaut.

Evaluation du volume de l'échantillon recueilli

Il faut connaître la quantité de matériel qui a précipité pour le pourcentage de solution saturée utilisée. Cette quantité doit ensuite être redissoute selon une proportion de 2 à 10 ml pour 100 mg de matériel. En fait, cette proportion dépend de la technique qui va suivre la précipitation. Si on envisage ensuite un échangeur d'ions, il n'y a pas de limite de volume. Une proportion de 10ml pour 100 mg de matériel est conseillée. Par contre, si on envisage un tamis moléculaire, le volume est limité à 20ml. Dans ce cas, il faudra ajuster la concentration pour disposer d'un volume convenable.

L'évaluation du volume de l'échantillon recueilli ne peut donc être réalisée lors de cette étape. Le système doit établir la description du nouveau mélange sans indiquer son volume. La description doit contenir une caractéristique spécifiant que le mélange n'est pas dissout.

Lorsque le système envisagera une autre technique pour le choix de la technique de purification la plus adéquate à appliquer au mélange, il conseillera à l'utilisateur une concentration telle que le volume du mélange soit adéquat à utiliser avec la technique.

S'il s'agit de la technique de l'échangeur d'ions, il conseillera une concentration de 10mg par ml et évaluera le volume du mélange pour une telle concentration.

S'il s'agit de la technique du tamis moléculaire, il conseillera une concentration la plus proche possible de 10mg par ml, tout en respectant la contrainte d'un volume inférieur ou à peine supérieur à 20ml.

Le système doit laisser la possibilité à l'utilisateur de refuser son conseil et de donner la concentration qu'il souhaite au mélange. L'utilisateur peut souhaiter travailler avec une concentration plus forte que la "concentration idéale" de 10mg par ml afin d'utiliser une colonne plus petite, manipuler un échantillon plus petit.

Evaluation de la durée de la précipitation

La durée d'une précipitation est approximativement d'une heure.

CHAPITRE 8. FORMALISATION

Nous formalisons ici la tâche de conception d'un plan de purification.

8.1. MODELISATION DU PROBLEME ET DE SA RESOLUTION [NIL71]

Nous avons décidé de modéliser le problème par une approche "espace d'états" qui est une approche assez classique en Intelligence Artificielle. La raison de ce choix est que cette approche convient bien à la modélisation des problèmes de planification en général et qu'elle semblait s'appliquer assez facilement à la modélisation de notre problème en particulier.

Cette modélisation met en oeuvre deux concepts importants : les états et les opérateurs. Un opérateur transforme un état en un autre état. Une solution au problème est une séquence d'opérateurs qui transforme un état initial en un état final.

Dans cette approche, un problème peut être décrit par le quadruplet ‹S, I, F, OP› où :

- S est un ensemble d'états
- I est un ensemble d'états initiaux représentant les données du problème et leurs propriétés. I est inclus dans S
- F est un ensemble d'états finals représentant la solution du problème et ses propriétés. F est inclus dans S
- OP est un ensemble d'opérateurs primitifs de transformation d'état. Un opérateur OPk, appartenant à OP, est défini de S dans S.

Apporter une solution au problème revient dès lors à déterminer une suite d'états intermédiaires Si, appartenant à S, tels que So appartient à I, Sf appartient à F et Si = OPk(Si-1), où OPk est un opérateur appartenant à OP et applicable à Si-1.

La solution du problème se compose d'une suite d'application d'opérateurs OPi1...OPin tels que

Sn = (OPin(OPin-1(...(OPi1(So))...)))

où So appartient à I et Sn appartient à F.

L'espace d'états est un graphe dans lequel un noeud représente un état et un arc un opérateur. Ce graphe représente tous les états possibles par application de tous les opérateurs applicables à partir d'un état initial.

La méthode de résolution consiste en une recherche dans ce graphe en procédant de manière incrémentale par essais-erreurs. C'est la méthode du "GENERATE AND TEST". On génère un nouvel état Si à partir d'un état Si-1, par application d'un opérateur OPk. On teste ensuite si cet état a les propriétés voulues, c'est-à-dire s'il est l'état final désiré ou s'il s'en rapproche strictement.

Nous allons illustrer la modélisation "espace d'états" sur un problème d'analyse syntaxique. Lorsqu'on manipule un langage, un problème fréquent consiste à savoir si une expression est conforme à la grammaire du langage ou non.

Supposons qu'une grammaire définisse qu'une phrase peut être :

- le symbole A suivi du symbole B
- le symbole A suivi d'une phrase
- une phrase suivie du symbole B
- une phrase suivie d'une autre phrase

S, l'ensemble des états correspondant à ce problème, est l'ensemble des "strings" formées des symboles A et B. Supposons que I, l'ensemble des états initiaux, contienne le "string" ABAABAB et que F soit l'ensemble composé du seul symbole P qui désigne une phrase. Les opérateurs peuvent être définis en termes des règles de réécriture suivantes :

$1AB$2 --> $1P$2
$1AP$2 --> $1P$2
$1PB$2 --> $1P$2
$1PP$2 --> $1P$2.

Ces opérateurs sont directement issus de la définition d'une phrase dans la grammaire. L'application de ces opérateurs donne un espace d'états assez grand. Une solution au problème pourrait être la suite d'états suivante, issue du graphe représentant l'espace d'états :

ABAABAB --> PAABAB --> PAPAB --> PPAB --> PPP --> PP --> P

Cependant, il apparaît dans la grammaire qu'une modélisation plus simple était possible. Il suffisait de ne définir qu'un seul opérateur :

A$B --> P.

L'espace d'états se réduisait alors à :

ABAABAB --> P.

Il est donc important de choisir une bonne modélisation.

En appliquant l'approche "espace d'états" à notre problème, nous obtenons la modélisation suivante :

- un état est la description d'un échantillon par ses caractéristiques présentée en 7.1.3.
- un opérateur est une technique chromatographique caractérisée par ses conditons d'application.

Si nous modélisons le problème sous forme d'un quadruplet ‹S, I, F, OP›, nous obtenons le quadruplet ‹E, EI, EF, T›, où :

- E est l'ensemble des échantillons
- EI est l'ensemble des échantillons initiaux. Cet ensemble contient des échantillons à purifier décrits par leurs caractéristiques
- EF est l'ensemble des échantillons finals. Il contient des échantillons qui sont les plus purs en fonction des connaissances du système ou dont le degré de purification satisfait l'utilisateur
- T est l'ensemble des techniques de purification actuellement disponibles pour le système, c'est-à-dire la chromatographie sur tamis moléculaire, la chromatographie sur échangeur d'ions et la précipitation isoélectrique.

Le plan de purification recherché pour un échantillon peut être décrit par :

$$Ef = (TEin(TEin-1(...(TEi1(Eo))...)))$$

où Ef appartient à EF, Eo appartient à EI et TEi appartient à T. TEj(Ei) (1 ‹= j ‹= 3) (1 ‹= i ‹= n) modélise la relation "s'applique à" entre une technique et un échantillon.

8.2. TECHNIQUE DE REPRESENTATION DES CONNAISSANCES [BRO85]

Nous utiliserons les règles de production comme technique de représentation des connaissances.

8.2.1. Architecture des systèmes de production

On appelle système de production un ensemble de règles de production et le mécanisme d'inférence qui est capable d'exécuter ces règles.

Le modèle de traitement des systèmes de production possède les trois composants majeurs du modèle procédural traditionnel : un programme, un exécuteur et des données.

L'architecture des systèmes de production est composée de :

- une mémoire de travail qui sert de base de données de symboles représentant des faits et des assertions concernant le problème. Les données sont des occurrences d'objets qui peuvent représenter soit des objets physiques ou des faits liés au domaine d'application, soit des objets conceptuels, tels que les buts, liés à la stratégie de résolution du problème
- un ensemble non ordonné de règles qui constituent le programme. Chaque règle est formée d'une partie condition (partie gauche), aussi appelée antécédent, décrivant les configurations de données pour lesquelles la règle est appropriée. Elle possède aussi une partie action (partie droite), aussi appelée conséquent, décrivant les instructions afin de changer les configurations de données. L'ensemble de règles est rangé dans la mémoire de production

- un moteur d'inférence qui exécute les règles. Il doit déterminer quelles règles sont applicables en fonction de la configuration des données et en choisir une pour l'appliquer. Le moteur met en oeuvre un mécanisme de contrôle appelé "cycle reconnaissance-action". Le contrôle dans un système de production est, de cette manière, basé sur de fréquentes réévaluations de l'état des données. C'est un traitement "data-driven" plutôt que "instruction-driven". Dans un même ordre d'idée, les règles ne peuvent se référencer entre elles explicitement. Il n'y a pas de transfert de contrôle entre règles d'un même niveau. Enfin, il y a séparation des connaissances dans les systèmes de production. La connaissance concernant le domaine se trouve dans les règles alors que le contrôle est assuré par le moteur d'inférence.

8.2.2. Le "matching" de règles avec des données

Pour déterminer quelles règles sont applicables en fonction de la configuration des données, le moteur d'inférence des systèmes de production utilise le "matching". Il s'agit de vérifier qu'un pattern faisant partie d'une règle "ressemble" à un élément (un fait ou un but) de la mémoire de travail.

Le "matching" le plus élémentaire vérifie qu'un pattern de littéraux est identique à un élément de la mémoire de travail. Le plus souvent, un "matching" plus souple est appliqué. Il peut par exemple permettre l'utilisation de variables dans le pattern. De cette manière, certaines portions du pattern restent non spécifiées. Lors du "matching", les variables sont liées, c'est-à-dire remplacées par une constante de la mémoire de travail, et cette liaison est gardée jusqu'à ce que le moteur abandonne la règle, après l'avoir appliquée ou non. Un ensemble de liaisons pour une règle est appelé une instanciation.

8.2.3. Stratégies de résolution et moteur d'inférence

Les règles dans les systèmes de production peuvent être appliquées dans deux directions. La direction utilisée dépend du type de stratégie de raisonnement employée par le moteur d'inférence.

En chaînage avant, les antécédents de la règle spécifient les combinaisons de faits que le moteur va tenter de faire "matcher" avec les éléments de la mémoire de travail. Le "matching" des antécédents ne peut avoir aucun effet de bord, dans le sens où il ne peut en aucune manière changer l'état de la mémoire de travail. Il s'agit d'un processus "bottom-up".

Lorsque les règles sont utilisées en chaînage arrière, les conséquents de la règle sont "matchés" avec le but initial ou des sous-buts, qui sont les conditions de règles ayant "matché" précédemment et qui n'étaient pas directement satisfaites. Il s'agit d'un processus "top-down".

Si la direction du chaînage est définie en termes du modèle "espace d'états", on peut concevoir la solution comme une recherche dans un espace d'états pour trouver un chemin entre une situation initiale et une situation finale.

Le chaînage avant progresse depuis la situation initiale jusqu'au but. Partant de ce qui est connu initialement, l'état courant des connaissances est utilisé pour construire une chaîne d'inférences jusqu'à ce qu'un but soit atteint ou qu'une solution apparaisse inatteignable.

Au contraire, le chaînage arrière part du but, le divise en sous-buts plus simples, jusqu'à ce qu'il en résulte un ensemble de sous-buts immédiatement atteignables ou les plus simples possibles. Les parties droites des règles représentent des buts à atteindre, alors que les parties gauches spécifient des conjonctions ou des disjonctions de sous-buts. Les buts directement atteignables correspondent à des éléments de la mémoire de travail.

8.2.4. Stratégies de contrôle

Les stratégies de contrôle déterminent quelles règles doivent participer au "matching" et quelle instanciation doit être choisie si plusieurs existent.

Différentes stratégies de contrôle existent et peuvent être mises en oeuvre en parallèle : la résolution de conflits, le filtrage et les méta-règles.

La résolution de conflit est un ensemble de principes pour sélectionner une instanciation parmi plusieurs possibles. Par exemple, un principe simple est que l'ordre dans lequel le moteur envisage les règles est celui dans lequel il les rencontre. Un autre principe est la sélection de la règle qui "matche" le plus souvent.

Le filtrage est une technique pour sélectionner quelles règles et quelles données participent au "matching". Le filtrage de règles réduit le nombre de règles qui peuvent "matcher" et le filtrage de données réduit le nombre de données que l'on peut envisager pour le "matching".

Les méta-règles sont une stratégie de contrôle de niveau supérieur. Elles spécifient explicitement comment appliquer les autres règles.

8.2.5. Représentation de nos connaissances par un système de production

De ce qui se trouve ci-dessus, il apparaît que les systèmes de production sont appropriés pour les problèmes dont la connaissance s'exprime aisément sous forme de règles, pour les programmes dont le contrôle s'avère conmplexe ou pour les programmes qui sont susceptibles d'être modifiés dans le futur.

Notre problème réunit ces trois conditions.

La connaissance relative aux techniques de purification est assez facilement exprimable sous forme de règles. Tout au long de nos entretiens avec les experts, il est apparu qu'ils avaient souvent tendance à exprimer, de manière intuitive, leur connaissance sous la forme situation-action. Il existe d'autres formes de représentation qui permettent également de faciliter le contrôle et la modifiabilité (forme clausale, par exemple) mais il était plus complexe d'exprimer la connaissance dans de telles formes.

La deuxième condition est moins évidente. Cependant, malgré le fait que nous nous soyons limités à trois techniques, le nombre de critères influançant le choix de ces techniques et leurs conditions d'application rendait la conception d'un programme, incluant la connaissance concernant le domaine et le contrôle d'exécution, assez compliquée.

Nous avons restreint le domaine à trois techniques et la connaissance concernant ces trois techniques doit encore être affinée. Le programme est donc destiné à être de nombreuses fois modifié dans le futur.

8.3. OUTIL DE REPRESENTATION DES CONNAISSANCES

Le langage LISP sera notre outil de représentation des connaissances.

Nous avions en fait le choix entre deux types de langages : les langages plus spécifiques à une approche algorithmique des problèmes, tels que PASCAL, et les langages de "manipulation de symboles". Parmi ceux-ci, LISP et PROLOG sont disponibles à l'Institut. Puisque nous avions décidé d'apporter à notre problème une solution de type système expert, les langages de "manipulation symbolique" s'imposaient.

Pour le choix entre LISP et PROLOG, deux arguments donnaient, au départ, la faveur à LISP. Le premier est que nous avions appris les concepts de base de PROLOG à l'occasion d'un cours de Techniques d'Intelligence Artificielle, en deuxième licence. Nous souhaitions également nous familiariser avec le type de programmation propre à LISP. Le second argument est que LISP offre une certaine facilité à représenter la connaissance dans une forme proche du langage naturel, sous forme de listes.

Des arguments plus déterminants sont ensuite intervenus. Tout d'abord, nous devons résoudre notre problème par mécanisme de chaînage avant, car nous ne disposons pas d'une description initiale de l'état final. Le moteur d'inférence PROLOG applique un mécanisme de chaînage arrière, partant de l'état final vers l'état initial. Il ne nous était donc pas possible de l'utiliser. Ensuite, nous avions choisi les systèmes de production comme technique de représentation des connaissances. Cette technique adopte une représentation des connaissances sous forme de règles de production. Il nous fallait un outil qui permette cette forme de représentation. PROLOG offre une représentation sous forme clausale basée sur la logique des propositions. Il était donc exclu de pouvoir l'utiliser.

Etant donné les disponibilités en matériel informatique du laboratoire, nous avions décidé de travailler sur micro-ordinateur de type PC. Deux dialectes de LISP étaient disponibles sur PC : XLISP et MULISP. Notre choix s'est porté sur MULISP.

MULISP est le résultat de nombreux efforts de conceptions pour une implémentation performante du LISP sur micro-ordinateur. L'objectif des concepteurs de MULISP a toujours été de fournir un noyau compact, pour permettre à l'utilisateur de développer ses propres environnements LISP. Parallèlement, les concepteurs se souciaient de rester compatible avec des dialectes plus connus. MULISP offre donc une librairie COMMONLISP et une librairie INTERLISP, pour implémenter les fonctions absentes du noyau MULISP. INTERLISP offre toutes les caractéristiques standards de LISP et un environnement élaboré qui inclut des facilités de "debugging", etc. COMMONLISP est un nouveau dialecte conçu pour fournir un standard LISP compatible avec un grand nombre d'ordinateurs. MULISP offre aussi des facilités pour le "debugging".

XLISP est un dialecte expérimental qui combine des caractéristiques de LISP avec des capacités d'extensions "orientées objet". Il a été implémenté pour permettre des expérimentations avec une programmation "orientée objet", sur de petits ordinateurs. Cette capacité a attiré notre attention lors d'un premier contact avec les dialectes disponibles. XLISP présentait cependant le défaut d'en être toujours au stade expérimental. Il n'était pas du tout certain que ce dialecte dipose de suffisamment de primitives LISP, alors que MULISP était doté d'une librairie COMMONLISP et d'une librairie INTERLISP. De plus, nous ne disposions que d'un manuel d'utilisation très restreint alors que nous étions des débutants en programmation LISP.

8.4. ARCHITECTURE DE LA MEMOIRE DE TRAVAIL

La mémoire de travail contiendra des assertions formalisant des concepts. Ces assertions seront représentées sous forme d'agrégats. En termes de représentation LISP, ces agrégats seront des listes. Le premier élément de chaque liste sera un atome désignant le nom du concept que l'assertion représente. Les éléments suivants seront des sous-listes de deux éléments. Le premier élément de chaque sous-liste désigne une caractéristique de la substance et l'élément suivant la(les) valeur(s) de cette caractéristique. Le second élément d'une sous-liste peut être une liste.

On trouvera aussi dans la mémoire de travail des assertions concernant l'état courant du processus de recherche.

Ces assertions seront modifiées par l'application de règles. De nouvelles assertions pourront être ajoutées.

La mémoire de travail elle-même est une liste de sous-listes, chaque liste étant une assertion.

Les assertions de la mémoire de travail formalisent les concepts ci-dessous.

8.4.1. Les substances

Les caractéristiques d'une substance sont celles présentées en 7.1.1. :

- le nom de la substance qui est une caractéristique obligatoire et identifiante

- la nature de la substance dont les valeurs les plus courantes sont "protéine", "enzyme" ou "sel"
- la masse moléculaire de la substance qui est exprimée en daltons
- le point isoélectrique qui est exprimé en pH
- le domaine de stabilité qui est exprimé en pH par deux valeurs : une borne inférieure et une borne supérieure. Ces deux valeurs sont contenues dans une liste
- le domaine de précipitation qui est exprimé en % par deux valeurs : une borne inférieure et une borne supérieure. Ces deux valeurs sont contenues dans une liste
- la stabilité dans le temps qui est exprimée en heures.

Aucune des caractéristiques d'une substance ne peut être modifiée par l'application d'une technique de purification.

8.4.2. Les mélanges

Les caractéristiques d'un mélange sont celles présentées en 7.1.2. :

- le nom du mélange qui est une caractéristique obligatoire et identifiante. Elle identifie une occurence de mélange. Cette caractéristique n'est pas modifiable par l'application d'une technique
- l'identification des substances du mélange qui est une caractéristique répétitive et décomposable. Les valeurs de cette caractéristique sont contenues dans une liste de sous-listes. Le premier élément de chaque sous-liste est le nom de la substance contaminante. Il fait référence à une occurence du composant du modèle appelé "substance". Le second

élément est la proportion de cette substance dans le mélange. La caractéristique ne peut être modifiée par l'application d'une technique. Elle peut cependant être modifiée lors de l'exécution de la tâche de détermination de l'échantillon à purifier. L'utilisateur peut ajouter ou retirer des substances au mélange. Il peut aussi en modifier les proportions.

8.4.3. Les échantillons

Un échantillon est un état de notre modélisation par l'approche "espace d'états". Il est décrit par un ensemble de caractéristiques qui déterminent le choix d'application d'une technique de purification.

Les caractéristiques d'un échantillon sont celles présentées en 7.1.3. :

- le nom du mélange qui est une caractéristique obligatoire. Elle identifie une occurence du composant du modèle appelé "mélange"
- le volume de l'échantillon qui exprimé en ml
- la quantité totale qui est exprimée en mg
- la concentration qui est exprimée en mg/ml. Si on connaît la quantité de substance dans l'échantillon, on peut obtenir la concentration par le rapport quantité de protéine/volume du mélange
- l'identification de la protéine à purifier qui est une caractéristique obligatoire. Elle identifie une occurence du composant du mélange appelé "substance"

- l'identification des substances contaminantes qui est une caractéristique répétitive et décomposable. Les valeurs de cette caractéristique sont contenues dans une liste de sous-listes. Le premier élément de chaque sous-liste est le nom de la substance contaminante. Il fait référence à une occurence du composant du modèle appelé "substance". Le second élément est la proportion de cette substance dans le mélange. La caractéristique peut être modifiée par l'application d'une technique à l'échantillon. L'application d'une technique peut modifier le pourcentage de contamination. Lorsque ce pourcentage est égal à 0, le contaminant ne fait plus partie des caractéristiques de l'échantillon.

Les caractéristiques nom du mélange et identification de la protéine à purifier ne peuvent être modifiées par l'application d'une technique à un échantillon. Les autres caractéristiques peuvent être modifiées.

8.4.4. Les gels

Les gels sont nécessaires à l'application de techniques de purification.

Les caractéristiques d'un gel sont :

- le type de gel qui est une caractéristique obligatoire et identifiante. Les différentes valeurs possibles sont celles des différents types de gel utilisés au laboratoire et présentés en 7.3.
- la marque dont les valeurs possibles sont "Pharmacia" et "Lkb"
- la charge dont les différentes valeurs possibles sont positive et négative

- le domaine de fractionnement qui est une caractéristique exprimée en daltons par deux valeurs : une borne inférieure et une borne supérieure
- le coefficient de non-séparation qui est une caractéristique répétitive.

Ces caractéristiques ne sont jamais modifiées par l'application d'une technique.

8.4.5. Les tampons

Les tampons sont nécessaires à l'application de techniques de purification.

Les caractéristiques d'un tampon sont :

- le nom du tampon qui est une caractéristique obligatoire et identifiante. Les différentes valeurs possibles sont celles des différents tampons utilisés au laboratoire et présentés en 7.3.
- le pK du tampon qui est exprimé en pH par une ou plusieurs valeurs. La (les) valeur(s) est (sont) contenue(s) dans une liste.

Ces caractéristiques ne sont jamais modifiées par l'application d'une technique.

8.4.6. Les colonnes

Les colonnes sont nécessaires à l'application de techniques de purification.

Les caractéristiques d'une colonne sont :

- le diamètre de la colonne qui est une caractéristique dont les différentes valeurs possibles sont 0.5 cm, 1 cm, 2 cm et 4 cm
- la hauteur de la colonne qui est une caractéristique dont la seule valeur possible actuellement est 80 cm
- la vitesse d'élution sur la colonne qui est une caractéristique dont les différentes valeurs possibles sont 2 ml/h, 15 ml/h ou 40 ml/h
- le volume maximum de l'échantillon que l'on peut déposer sur la colonnne et dont les différentes valeurs possibles sont 0.25 ml, 1 ml, 3.5 ml, 20 ml.

Ces caractéristiques ne sont jamais modifiées par l'application d'une technique.

La description des colonnes n'est nécessaire que pour la technique du tamis moléculaire. Pour l'application de la technique de l'échangeur d'ions, le système ne doit mémoriser aucune description de colonne.

8.4.7. Les techniques

Une technique est un opérateur de notre modélisation par l'approche "espace d'états". Elle est décrite par un ensemble de caractéristiques qui sont ses conditions d'application.

La formalisation du concept "technique" et de la relation "s'applique à" entre une technique et un échantillon donne le composant du modèle appelé "technique". Les caractéristiques peuvent faire référence à des concepts qui seront eux-mêmes représentés par des agrégats. C'est par exemple le cas de la caractéristique "type de gel" qui fait référence à un agrégat représentant un gel et ses caractéristiques.

Les caractéristiques d'une technique sont :

- le nom de la technique qui est une caractéristique obligatoire et identifiante. Elle peut prendre les valeurs "tamis moléculaire", "échangeur d'ions" ou "précipitation isoélectrique"
- le diamètre de la colonne qui est une caractéristique facultative. Elle n'existe que pour les techniques dont l'application nécessite une colonne. Il s'agit de l'échangeur d'ions et du tamis moléculaire. Si la technique est le tamis moléculaire, cette caractéristique prendra la valeur attachée à la colonne choisie en fonction du volume de l'échantillon. Cette valeur est une caractéristique de l'agrégat formalisant une colonne. Si la technique est l'échangeur d'ions, le diamètre de la colonne prendra une valeur calculée en fonction de caractéristiques de l'échantillon et d'un choix de l'utilisateur
- le volume de la colonne qui est une caractéristique facultative. Elle n'existe que pour le tamis moléculaire.

- la hauteur de la colonne qui est une caractéristique facultative. Elle n'existe que pour les techniques dont l'application nécessite une colonne. Il s'agit de l'échangeur d'ions et du tamis moléculaire. Si la technique est le tamis moléculaire, cette caractéristique prendra la valeur attachée à la colonne choisie en fonction du volume de l'échantillon. Cette valeur est une caractéristique de l'agrégat formalisant une colonne. Si la technique est l'échangeur d'ions, la hauteur de la colonne prendra une valeur choisie par l'utilisateur
- le type de gel qui est une caractéristique facultative. Elle n'existe que pour les techniques échangeur d'ions et tamis moléculaire. Cette caractéristique référence une occurence d'un agrégat formalisant un gel. La valeur de cette caractéristique est déterminée en fonction de l'échantillon à purifier et de la technique appliquée
- le volume de gel est une caractéristique facultative. Elle n'existe que pour les techniques échangeur d'ions et tamis moléculaire. La valeur de cette caractéristique est déterminée en fonction de l'échantillon à purifier et d'autres caractéristiques de la technique appliquée, lorsqu'il s'agit de l'échangeur d'ions. Pour le tamis moléculaire, il n'est pas nécessaire de connaître le volume de gel avant l'application de la technique. On remplit la colonne conseillée de gel
- le nom du tampon qui est une caractéristique facultative et répétitive. Elle n'existe que pour les techniques échangeur d'ions et tamis moléculaire. Cette caractéristique référence une occurence d'un agrégat formalisant un tampon. La valeur de cette caractéristique est déterminée en

fonction de l'échantillon à purifier et de la technique appliquée
- le type de décrochage qui est une caractéristique facultative. Elle n'existe que pour l'échangeur d'ions. Elle est déterminée en fonction d'un choix de l'utilisateur. Les valeurs possibles sont le décrochage en étapes ou par gradient
- le type d'élution qui est une caractéristique facultative. Elle n'existe que pour l'échangeur d'ions. Elle est déterminée en fonction des caractéristiques de l'échantillon et des conditons d'application de la technique. Les valeurs possibles sont la modification de la force ionique ou la modification de la force ionique et du pH
- la durée d'application qui est une caractéristique obligatoire. Elle est déterminée en fonction des caractéristiques de l'échantillon et des conditons d'application de la technique. Les valeurs sont exprimées en heures. Dans un premier temps, les valeurs de cette caractéristique ne seront que très approximatives, la connaissance nécessaire à leur détermination n'étant pas suffisante
- la perte d'activité de la protéine à purifier qui est une caractéristique facultative. Elle n'existe que pour l'échangeur d'ions. Elle est déterminée en fonction des caractéristiques de l'échantillon et des conditons d'application de la technique. Les valeur sont exprimées en %. Dans un premier temps, les valeurs de cette caractéristique ne seront que très approximatives, la connaissance nécessaire à leur détermination n'étant pas suffisante. De même pour le tamis moléculaire et pour la précipitation, cette caractéristique n'existe pas par manque de connaissance

- le volume de l'agent de précipitation qui est une caractéristique facultative. Elle n'est définie que pour la précipitation. Elle est déterminée par les caractéristiques de l'échantillon et est exprimée en ml.

Un agrégat "technique" est créé lors du choix de l'application d'une technique à un échantillon. Cet agrégat comprend, dans un premier temps, la caractéristique "nom de la technique". Il est ensuite complété par les autres caractéristiques au fur et à mesure que les conditions d'application de la technique sont envisagées. La technique est ensuite conseillée à l'utilisateur, puis l'agrégat est retiré de la base de connaissance. Un autre agrégat "technique" est créé et ainsi de suite jusqu'à obtention de tout un plan de purification. Le composant technique est de cette manière la formalisation du concept technique et de la relation "s'applique à" entre une technique et un échantillon. Cela est réalisé par les règles qui contiennent la connaissance relative aux techniques et à leurs conditions d'application.

8.5. ARCHITECTURE DE LA BASE DE REGLES

La base de connaissances contiendra les règles concernant les trois techniques de purification. Ces règles sont la mise en oeuvre des concepts présenté en [7.3.].

En termes de représentation LISP, une règle est une liste composée de l'identification de la règle et de deux sous-listes : une liste d'antécédents et une liste de conséquents. Le premier élément de la sous-liste des antécédents est l'atome "IF" et les éléments suivants sont des listes, chaque liste étant un antécédent. Le premier

élément de la sous-liste des conséquents est l'atome "THEN" et les éléments suivants sont des listes, chaque liste étant un conséquent. L'utilisation des atomes "RULE", "IF" et "THEN" facilite la compréhension des règles. Toutes les règles auront donc la forme :

```
(rule <nom>
   (IF (<antécédent1>)
       (<antécédent2>)
           ...
       (<antécédentn>)
   )
   (THEN (<conséquent1>)
         (<conséquent2>)
              ...
         (<conséquentn>)
   )
)
```

Nous utilisons le concept LISP de liste d'associations pour représenter la base de règles. En LISP, une liste d'associations est une liste de sous-listes, dans laquelle le premier élément de chaque sous-liste est une clé et le second élément est la valeur de cette clé. Notre base de connaissances est une liste de sous-listes. La clé de chaque sous-liste est le nom de la règle, permettant de sélectionner la règle et de l'obtenir. Sa valeur est la règle proprement dite.

La base contiendra deux groupes de règles. Le premier groupe de règle réalise la tâche de choix d'application d'une technique de purification. Le second groupe assure la tâche d'évaluation de l'application de la technique.

8.5.1. Règles de choix d'application d'une technique

Ce premier groupe de règles contient la connaissance nécessaire au choix d'une technique de purification. Ce choix est réalisé en fonction des caractéristiques de l'échantillon. La conséquence de ce choix est la création dans la base de faits d'un agrégat "technique" et de la caractéristique "nom" de cet agrégat.

Ce même groupe de règles met au point les conditions d'application de la technique sur base des caractéristiques de l'échantillon, des conditions d'application déjà précisées et de choix spécifiés par l'utilisateur. Il en résulte que l'agrégat "technique" est complété par les différentes caractéristiques correspondant aux conditions d'application envisagées.

8.5.2. Règles d'évaluation de l'application de la technique

Ces règles continuent la mise au point des conditions d'application de la technique non encore spécifiées par le premier groupe. Ce second groupe de règles contient également la connaissance relative à l'évaluation des conséquences de l'application d'une technique à un échantillon. L'évaluation s'effectue sur base des caractéristiques de l'échantillon avant purification et des conditions d'application de la technique envisagée. Il en résulte un nouvel agrégat "échantillon" qui n'est autre que l'agrégat existant avant l'application de la technique, dont les caractéristiques ont été modifiées.

8.6. LE MOTEUR D'INFERENCE

Notre moteur d'inférence appliquera un "pattern matching" étendu, c'est-à-dire que nos règles contiendront des variables. Le moteur d'inférence sera en outre capable d'appliquer des opérateurs contenus dans les règles. Ces opérateurs seront des primitives prédéfinies dans le langage ou par les concepteurs du système. Ceci sera détaillé dans l'implémentation.

Nous avons opté pour une stratégie de chaînage avant. Nous n'avions en fait pas le choix, puisque dans la description initiale de notre problème, nous ne disposons pas d'une description de l'état final souhaité. Notre système doit donc nécessairement partir de l'état initial vers le but.

8.7. STRATEGIE DE CONTROLE

La stratégie de contrôle dans notre système est réalisée par les trois moyens présentés brièvement en [8.2.4.].

En ce qui concerne la résolution de conflit, l'ordre dans lequel le moteur envisagera les règles est celui dans lequel il les rencontre. La première de la liste qui "matche" sera donc appliquée. Du point de vue du concepteur, il est important de tenir compte de ce principe dans la manière de formuler les règles. Il faut formuler les antécédents de manière telle que certaines règles ne puissent "matcher" et être appliquées que si d'autres ont été appliquées précédemment. Nous devons aussi veiller à

l'ordre dans lequel il place les règles dans la liste qui sera parcourue par le moteur.

Le deuxième moyen de contrôle présenté en [8.2.4.] est le filtrage. Dans notre système, il est réalisé par le fait qu'au moment du lancement de la tâche de conception d'un plan de purification, seules les données propres à l'échantillon à traiter sont chargées dans la mémoire de travail. Il s'agit d'un filtrage sur les données.

Nous complétons enfin le contrôle de l'exécution des règles par un groupe de règles d'un niveau supérieur aux règles représentant la connaissance. Il s'agit de méta-règles, qui constituent la base de méta-règles. Ce sont des règles qui déterminent l'ordre dans lequel les règles d'un niveau inférieur doivent être envisagées.

En termes de représentation LISP, une méta-règle a la même structure qu'une règle de la base de connaissances. La base de méta-règles est un liste de sous-listes, où chaque sous-liste est une méta-règle.

Le moteur va exécuter les méta-règles. Dans certains cas, les antécédents de ces méta-règles vont "matcher" des règles à appliquer dans la base de connaissances. Les conséquents consistent alors à appliquer une primitive d'exécution de règles du moteur d'inférence aux règles qui ont "matché" avec les antécédents, et cela dans un certain ordre. Dans d'autres cas, les méta-règles travaillent sur la mémoire de travail, tout comme les règles de la base de connaissances, mais pour assurer le contrôle.

Les méta-règles permettent un transfert de contrôle entre règles, mais ce transfert a lieu entre règles de niveaux différents.

Cette stratégie de contrôle nous a semblé plus aisée à mettre en oeuvre que des prévisions sur l'ordre dans lequel les règles doivent se trouver dans la base de connaissances. Elle permet aussi d'augmenter l'efficacité du système.

CHAPITRE 9. IMPLEMENTATION

9.1. IMPLEMENTATION DU MOTEUR D'INFERENCE [WIN84]

Comme nous l'avons précisé lors de l'étape de formalisation, nous représentons les connaissances relatives aux techniques de purification sous forme de règles de production. Nous adoptons une stratégie de résolution en chaînage avant ou "forward chaining".

Bien que Lisp ne comporte pas de tel processus, il est relativement aisé de définir des fonctions constituant le moteur d'inférence.

Cette section spécifie les concepts et les procédures utilisées par le moteur d'inférence.

9.1.1. Manipulation des faits

Les assertions de la mémoire de travail sont représentés sous forme d'expressions symboliques. Nous utilisons la structure de liste. Lors de l'exécution de la tâche de construction d'un plan de purification, une liste d'assertions représente la base des faits, c'est-à-dire la mémoire de travail.

Pour manipuler la liste d'assertions représentant la base de faits, nous définissons deux fonctions, l'une permettant d'ajouter une assertion à la liste d'assertions, l'autre permettant d'enlever une assertion de la liste d'assertions :

add-assertion

arguments : assertions et assert
assert est une assertion à ajouter à la liste d'assertions assertions qui représente la base de faits
résultat : assert
effet de bord : assertions
si assert appartient à assertions, assertions est inchangée, sinon assert est ajoutée à assertions ; la fonction renvoie asssert

remove-assertion
arguments : assertions et assert
assert est une assertion à enlever de la liste d'assertions assertions qui représente la base de faits
résultat : assert
effet de bord : assertions
si assert appartient à assertions, assert est enlevée d'assertions, sinon assertions est inchangée. La fonction renvoie assert

9.1.2. "Pattern matching" symbolique

9.1.2.1. Description du "pattern matching" symbolique

Le "pattern matching" est le processus de comparaison d'expressions symboliques qui permet de déterminer si deux expressions symboliques sont identiques.

Une expression symbolique est une assertion ou un pattern. Une assertion est un fait établi de la mémoire de travail. Un pattern est une expression à valider par comparaison avec une assertion. La procédure de "pattern matching" compare un pattern et une assertion.

Dans notre implémentation, les expressions symboliques sont représentées par des listes d'éléments. La procédure de comparaison avance en même temps dans le pattern et l'assertion, s'assurant que les éléments correspondent deux à deux.

Les éléments d'une assertion sont des atomes et/ou des listes d'atomes. Une liste d'atomes contient des atomes et/ou des listes d'atomes.

Les éléments d'un pattern peuvent être des atomes ou des listes d'atomes. Le pattern "matche" alors avec l'assertion si les éléments placés dans la même position sont identiques :

```
(match '(albumine proteine (pm 67000))
       '(albumine proteine (pm 67000))
)
t

(match '(hemoglobine proteine) '(hemoglobine enzyme))
nil.
```

La procédure renvoie "t" s'il y a "matching" et "nil" sinon.

Un pattern peut contenir des symboles particuliers qu'une assertion ne peut contenir : '?' et '+'. Le symbole '?' a le privilège de "matcher" avec tout atome et toute liste d'atomes. Le symbole '+' peut "matcher" avec un ou plusieurs atomes et plusieurs listes d'atomes. Le symbole '+' peut également "matcher" avec rien, c'est-à-dire que la procédure peut ignorer la présence du symbole '+' dans le pattern :

```
(match '(hemoglobine ? a purifier)
       '(hemoglobine proteine a purifier))
t

(match '(hemoglobine + purifier)
       '(hemoglobine proteine a purifier))
t

(match '(hemoglobine + proteine a purifier +)
       '(hemoglobine proteine a purifier))
t
```

Comme nous l'avons annoncé lors de la formalisation, nous utilisons un "pattern matching" étendu, qui permet l'utilisation de variables dans le pattern.

Le pattern peut contenir des listes-variable. Une liste-variable est une liste de deux éléments, un indicateur de variable et une variable. Il existe quatre indicateurs de variable : '>>', '<<', '>+' et '<+'. Les listes-variable permettent d'utiliser les valeurs associées aux symboles plutôt que les symboles eux-mêmes dans le "matching". La syntaxe d'une liste-variable est :

(<indicateur de variable> <variable>)

L'utilisation de variables permet d'associer des symboles à des valeurs si le "matching" réussit. L'indicateur de variable '>>' permet d'associer un symbole représentant une variable à une valeur contenue dans l'assertion. Lorsque le "matching" réussit, la variable du pattern est associée à l'atome ou la liste d'atomes qui lui correspond dans l'assertion. L'atome ou la liste d'atomes de l'assertion devient la valeur de la variable du pattern. La liste constituée du symbole représentant la variable et de la valeur associée est rangée dans une liste

d'associations. Dans l'utilisation que nous faisons de ce concept Lisp, une sous-liste (variable valeur) forme une sous-liste (clé valeur) d'une liste d'associations. Lorsqu'une variable du pattern est associée à une valeur de l'assertion, la sous-liste qu'elles forment est ajoutée à la liste d'associations si la variable n'est la clé d'aucune sous-liste de la liste d'associations. Si une sous-liste de la liste d'associations a déjà pour clé la variable du pattern, la valeur qui y était associée est remplacée dans la liste d'associations par la valeur correspondant à la variable du pattern dans l'assertion. La liste d'associations contient l'ensemble des liaisons réalisées par le "matching". La procédure de "matching" renvoie la liste d'associations qui lui est associée lorsqu'il y a "matching" et que celle-ci n'est pas vide, renvoie "t" s'il y a "matching" et que la liste d'associations est vide, et renvoie "nil" s'il n'y a pas "matching" :

```
(match '((>> proteine) proteine a purifier)
       '(hemoglobine proteine a purifier))
((proteine hemoglobine)).
```

Comme il est utile d'associer des valeurs aux variables, il est intéressant d'utiliser les valeurs dans le "pattern matching". L'indicateur de variable '<<' permet d'utiliser la valeur d'un symbole représentant une variable plutôt que le symbole lui-même. Il indique que la variable doit être remplacée par sa valeur pour réaliser le "matching". La procédure de "pattern matching" consulte la liste d'associations qui lui est associée pour connaître la valeur par laquelle la variable doit être remplacée. Si la variable est la clé d'une sous-liste de la liste d'associations, pour réaliser le "matching", elle est remplacée par la valeur qui lui est associée. Si la variable n'est la clé d'aucune sous-liste de la liste d'associations, elle est remplacée par nil :

```
(match '((>> proteine) + (<< proteine))
       '(hemoglobine est identique a hemoglobine))
((proteine hemoglobine))
```

L'indicateur de variable '>+' permet d'associer plusieurs éléments de l'assertion à la variable du pattern. La valeur de la variable est une liste d'atomes. Lorsque le "matching" réussit, les éléments de l'assertion qui correspondent à la variable sont ajoutés à la liste d'atomes qui constitue la valeur de la variable dans la liste d'associations, pour autant que la variable soit la clé d'une sous-liste de la liste d'associations. Si aucune sous-liste de la liste d'associations n'a pour clé la variable du pattern, une nouvelle sous-liste est constituée dans la liste d'associations. Elle se compose de la variable et de la liste d'atomes contenant les éléments de l'assertion qui correspondent à la variable :

```
(match '((>+ substance) a purifier)
       '(hemoglobine proteine a purifier))
((substance (hemoglobine proteine))).
```

L'indicateur de variable '<+' permet de remplacer une variable du pattern par les éléments qui constituent la liste associée à cette variable dans une sous-liste de la liste d'associations :

```
(match '((>+ substance) egale (<+ substance))
       '(hemoglobine proteine egale hemoglobine proteine))
((substance (hemoglobine proteine)))
```

Les indicateurs de variable '>>' et '<<' sont complémentaires : '>>' associe une valeur à une variable après le "matching" et '<<' prend la valeur de la variable

avant le "matching". Il en est de même pour les indicateurs de variable '>+' et '<+'.

Si un élément du pattern est une liste, elle peut être une liste d'atomes. Comme nous l'avons déjà dit, la comparaison se réalise alors avec la liste d'atomes correspondante dans l'assertion : les éléments des deux listes doivent correspondre deux à deux. Un élément liste du pattern peut également contenir les symboles particuliers '?' et '+' et des listes-variable (>> var), (<< var), (>+ var) et (<+ var). Dans ces cas, la comparaison se réalise par le "matching" de la liste du pattern avec l'élément qui lui correspond dans l'assertion.

9.1.2.2. Spécification des fonctions utiles au "pattern matching" symbolique

Nous spécifions à présent les fonctions qui permettent de réaliser le "pattern matching". Le code du programme est présenté dans l'annexe A.

Pour manipuler les listes-variable, nous définissons deux fonctions, l'une permettant d'accéder à l'indicateur de la liste, l'autre permettant d'accéder à la variable de la liste :

```
pattern-indicator
  argument : lst
        lst est une liste-variable
  résultat : indicator
        la fonction renvoie indicator, le premier élément de
        la liste-variable lst

pattern-variable
  argument : lst
```

lst est une liste-variable

résultat : var

la fonction renvoie var, le deuxième élément de la liste-variable lst

Pour manipuler la liste d'associations associée à la procédure de "pattern-matching", nous définissons quatre fonctions.

La première permet de construire une liste d'associations en y déposant une sous-liste (variable valeur) :

shove-pair

arguments : var, val et lstasso

var est un symbole ; val est un atome ou une liste d'atomes ; (var val) est une sous-liste à ajouter à la liste d'associations lstasso

résultat : lstasso'

la fonction renvoie lstasso'. Si aucune sous-liste de la liste d'associations lstasso n'avait var pour clé, la sous-liste (var val) est ajoutée à lstasso pour former lstasso' :

lstasso' = lstasso + {(var val)}.

Si var était la clé d'une sous-liste de la liste d'associations lstasso, la valeur valanc qui était associée à var est remplacée par la valeur val pour former lstasso' :

lstasso' = lstasso - {(var valanc)} + {(var val)}.

Nous définissons deux fonctions permettant de construire une liste d'associations en ajoutant respectivement un élément et plusieurs éléments à la liste d'atomes constituant la valeur d'une variable dans une liste d'associations initiale :

shove-group

arguments : var, val et lstasso

var est un symbole ; val est un atome ou une liste d'atomes, élément à ajouter à lval, (var lval) étant une sous-liste de la liste d'associations lstasso

résultat : lstasso'

la fonction renvoie lstasso'. Si aucune sous-liste de la liste d'associations lstasso n'a var pour clé, c'est-à-dire que (var lval) n'existe pas, la sous-liste (var (val)) est ajoutée à lstasso pour former lstasso' :
lstasso' = lstasso + {(var (val))}.
Si (var lval) existe avec lval = (v1 v2 ... vn), val est ajoutée à lval :
lstasso' = lstasso - {(var lval)} + {(var lval')}
avec lval' = (v1 v2 ... vn val).

shove-groups

arguments : var, lstval et lstasso

var est un symbole ; lstval est une liste d'atomes, éléments à ajouter à lval, (var lval) étant une sous-liste de la liste d'associations lstasso

résultat : lstasso'

la fonction renvoie lstasso'. Si aucune sous-liste de la liste d'associations lstasso n'a var pour clé, c'est-à-dire que (var lval) n'existe pas, la sous-liste (var (val1 val2 ... valn)) avec lstval = (val1 val2 ... valn) est ajoutée à lstasso pour former lstasso' :
lstasso' = lstasso + {(var (val1 val2 ... valn))}.
Si (var lval) existe avec lval = (v1 v2 ... vn),
lstasso' = lstasso - {(var lval)} + {(var lval')}
avec lval' = (v1 v2 ... vn val1 val2 ... valn).

La quatrième fonction permet d'obtenir la valeur d'une variable de la liste d'associations :

pull-value

arguments : var et lstasso

var est un symbole représentant une variable et lstasso est une liste d'associations

résultat : val

la fonction renvoie val. Si var est une clé d'une sous-liste de la liste d'associations lstasso, alors val est la valeur associée à la clé var. Sinon val est égal à "nil".

Le "pattern matching" est assuré par la fonction "match" :

match

arguments : pattern, assertion, lstasso

pattern et assertion sont deux expressions symboliques, respectivement un pattern et une assertion au sens ils ont été définis en 9.1.2.1. lstasso est une liste d'associations

résultat : lstasso', "t" ou "nil"

la fonction compare pattern et assertion pour vérifier s'ils "matchent" au sens où le "pattern matching" a été défini en 9.1.2.1. S'il y a "matching", la fonction renvoie lstasso' si cette liste d'associations n'est pas vide : lstasso' est égale à lstasso complétée des sous-listes éventuellement constituées ou modifiées par le "matching". S'il y a "matching", la fonction renvoie "t" lorsque lstasso' est vide. S'il n'y a pas "matching", la fonction renvoie "nil".

9.1.3. Les patterns-étendus

Le processus de "pattern-matching" permet de "matcher" une assertion de la mémoire de travail et un pattern quelconque.

Cependant, comme nous l'avons présenté lors de la formalisation, le moteur d'inférence exécute les méta-règles. L'exécution d'une méta-règle permet de déterminer les règles de connaissances à appliquer. Pour réaliser cela, le moteur d'inférence doit réaliser du "pattern-matching" sur la base de connaissances des règles.

Pour implémenter ce "pattern-matching" particulier, nous définissons les "patterns-étendus", qui sont des listes de deux sous-listes. La première sous-liste est égale à (rule) et la seconde est un pattern. La sous-liste (rule) indique au moteur d'inférence que le "matching" doit se réaliser entre la base de règles et le pattern spécifié dans la seconde sous-liste.

9.1.4. Les opérations

Une opération est une expression symbolique qui impose au moteur l'application d'une fonction avec des arguments déterminés.

Une opération a pour syntaxe :

```
((ope ‹nom de fonction›) (res ‹résultat attendu›)
                          ‹arg 1› ‹arg 2› ... ‹arg n›
).
```

L'élément (ope ‹nom de fonction›) de l'opération est une liste-fonction. Elle comprend deux éléments, un

indicateur de fonction 'ope' et un identificateur de fonction prédéfinie du langage ou définie par l'utilisateur.

Une opération peut être une opération-test. Elle est alors destinée à appliquer une fonction et à évaluer ensuite si le résultat renvoyé par la fonction correspond au résultat attendu. L'élément (res ‹résultat attendu›) a la forme (res ‹valeur›). Le résultat attendu est une valeur précise (nil, t, 20 par exemple) ou une valeur non précisée différente de nil. Dans ce cas, le résultat attendu est le symbole "nonil".

Une opération peut être une opération-calcul. Elle est alors destinée à appliquer une fonction, à associer le résultat renvoyé par la fonction à un symbole et, éventuellement, à évaluer si le résultat renvoyé par la fonction correspond à un résultat attendu. L'élément (res ‹résultat attendu›) a la forme (res ‹liste-variable› ‹valeur›). La liste-variable a pour indicateur le symbole '›>'. Le résultat renvoyé par la fonction est associé au symbole de la liste-variable représentant une variable. L'expression ‹valeur› est le résultat attendu. Elle est comparée au résultat renvoyé par la fonction. Elle peut être une valeur précise (nil, t, 20 par exemple) ou une valeur non précisée différente de nil. Dans ce cas, le résultat attendu est le symbole "nonil". L'expression ‹valeur› est facultative.

Un argument ‹arg i› peut être un atome, une liste-variable d'indicateur '‹<' ou '‹+', une opération-test ou une liste d'arguments.

Pour manipuler les opérations, nous définissons plusieurs fonctions :

operation-indicator
  argument : lst
      lst est une liste-fonction
  résultat : indicator
      la fonction renvoie indicator, le premier élément de la liste-fonction lst ; indicator est égal à "ope"

operation-function
  argument : lst
      lst est une liste-fonction
  résultat : function
      la fonction renvoie function, le deuxième élément de la liste-fonction lst

replace-variables
  arguments : lstargu et lstasso
      lstargu est une liste d'éléments. Un élément est un atome, une liste-variable d'indicateur '<<' ou '<+', une opération-test ou une liste d'éléments ; lstasso est une liste d'associations
  résultat : lstargu'
  effets de bord : effets de bord des opérations
      la fonction renvoie lstargu', une liste d'arguments égale à lstargu dans laquelle les arguments ont été évalués. Si l'argument est un atome, il est évalué à lui-même. Si l'argument est une liste-variable, la liste est remplacée par la valeur qui lui correspond dans la liste d'associations lstasso. S'il ne lui en correspond pas, la liste est remplacée par "nil". S'il s'agit d'une opération-test, elle est appliquée et remplacée par la valeur qu'elle renvoie. Dans ce cas, les fonctions appliquées peuvent éventuellement avoir des effets

de bord. S'il s'agit d'une liste, les éléments qu'elle contient sont évalués.

apply-operation
  arguments : operation et lstasso
      operation est une opération et lstasso est une liste d'associations
  résultat : lstasso', "t" ou "nil"
  effets de bord : effets de bord de la fonction appliquée
      apply-operation applique une fonction. Ses effets de bord sont les effets de bord de la fonction appliquée. Si l'opération est une opération-test, apply-operation compare le résultat obtenu au résultat attendu. Si les résultats correspondent, apply-operation renvoie "t", sinon renvoie "nil". Si l'opération est une opération-calcul, la valeur attendue comme résultat est comparée au résultat renvoyé par la fonction. S'ils correspondent, l'opération réussit. S'il n'y a pas de valeur attendue, l'opération réussit. Si l'opération réussit, le résultat renvoyé par la fonction est associé à la variable et cette sous-liste est ajoutée à la liste d'associations lstasso pour former la liste d'associations lstasso' que renvoie apply-operation. Si l'opération ne réussit pas, apply-operation renvoie "nil".

9.1.5. Les accès

Lors de la formalisation, nous avons introduit la représentation des concepts tels que ceux de technique, d'échantillon. Ce sont des assertions dans la liste des assertions représentant la base de faits. Ils possèdent des attributs qui les caractérisent. Un attribut peut être identifiant pour l'assertion. Nous utilisons cet attribut

identifiant d'une assertion pour accéder à cette assertion dans la liste d'assertions. Un accès est une expression symbolique qui permet au moteur d'accéder à une assertion par son identifiant afin d'ajouter ou de modifier certains attributs de l'assertion.

Un accès a pour syntaxe :
((id <identifiant>) <action1> <action2> ... <actionn>)

L'élément (id <identifiant>) de l'accès est une liste-identification. Elle comprend deux éléments, un indicateur d'identifiant 'id' et un attribut identifiant d'une assertion.

Un élément <actioni> impose au moteur d'ajouter ou de modifier un attribut de l'assertion identifiée. Il a la forme <action clé liste-de-valeurs> avec liste-de-valeurs = (elem1 elem2 ... elemn). L'action peut être un ajout ou une modification : action est égal à 'a' ou 'm'. La clé désigne le nom de l'attribut dans l'assertion. Si l'action est un ajout, les éléments formant la liste de valeurs sont ajoutés à la liste des éléments formant la valeur de l'attribut dans l'assertion, pour autant que l'attribut existe. Si l'attribut n'existe pas, il est créé avec pour valeur la liste de valeurs. Si l'action est une modification, la liste de valeurs comporte un seul élément qui devient la valeur de l'attribut dans l'assertion. Si l'attribut n'existe pas, il est créé avec l'élément pour valeur. Un élément de la liste de valeurs peut être un atome, une liste-variable d'indicateur '<<' ou '<+', une opération-test ou une liste d'éléments.

Pour manipuler les accès, nous définissons plusieurs fonctions :

access-indicator
  argument : lst
      lst est une liste-identification
  résultat : indicator
      la fonction renvoie indicator, le premier élément de la liste-identification lst ; indicator est égal à "id"

access-identificator
  argument : lst
      lst est une liste-identification
  résultat : identificator
      la fonction renvoie identificator, le deuxième élément de la liste-identification lst

apply-access
  arguments : access, lstasso et asserts
      access est un accès ; lstasso est une liste d'associations ; asserts est la liste d'assertions représentant la base de faits
  résultat : assert
  effets de bord : asserts
      apply-access tente de réaliser un accès à l'assertion de asserts identifiée dans l'accès. Si aucune assertion de asserts n'est identifiée par l'accès, apply-access ajoute une nouvelle assertion à asserts, réalise les actions spécifiées dans access et renvoie assert, l'assertion créée. Si l'accès à l'assertion est réalisé, apply-access exécutent les actions spécifiées dans access et renvoie assert, l'assertion modifiée par les actions. L'exécution des actions modifie asserts

build-assertion
  arguments : id, oldatt et newatt
      id est un identifiant d'une assertion assert ; oldatt est une liste de sous-listes composées d'un attribut de assert et de sa valeur ; newatt est une liste d'actions d'un accès
  résultat : assert'
      build-assertion construit une assertion assert' ayant comme premier élément id, suivi des attributs de oldatt modifiés par les actions de newatt et des attributs appartenant à newatt sans appartenir à oldatt. Si oldatt est vide, assert' est construite avec id suivi des attributs spécifiés dans newatt. Si newatt est une liste vide, assert' est l'assertion construite avec id et oldatt. Si newatt et oldatt sont vides, assert' est une assertion contenant un seul élément, id.

9.1.6. Manipulation des règles

La représentation des règles choisie lors de la formalisation a pour syntaxe :

```
(rule <name>
      (if <antécédent 1>
          <antécédent 2>
          ...
          <antécédent n>
      )
      (then <conséquent 1>
            <conséquent 2>
            ...
            <conséquent n>
      )
)
```

Les antécédents et les conséquents sont des expressions symboliques. Lorsqu'il applique une règle, le moteur d'inférence évalue d'abord les antécédents. Si ceux-ci réussissent, il exécute alors les conséquents. C'est dans ce cas que la règle réussit. Par contre, si un antécédent échoue, les éventuels antécédents suivants ne sont pas évalués, le moteur n'exécute pas les conséquents et la règle échoue.

9.1.6.1. Un antécédent

Un antécédent est soit un pattern, soit une opération.

Si l'antécédent est un pattern, il réussit si le moteur réussit à "matcher" le pattern avec une ou plusieurs assertions de la liste d'assertions représentant la base de faits.

Si l'antécédent est une opération-test, le résultat attendu est une valeur précise et l'antécédent réussit si le résultat renvoyé par la fonction est égal à cette valeur précise, ou une valeur non précisée et le résultat renvoyé doit être différent de "nil" pour que l'antécédent réussisse. Si l'antécédent est une opération-calcul, l'absence de l'expression ‹valeur› signifie que l'antécédent réussit toujours, tandis que sa présence impose des conditions de réussite analogues à celles d'une opération-test.

9.1.6.2. Un conséquent

Un conséquent est un pattern, une opération ou un accès.

Si le conséquent est un pattern, le moteur doit ajouter l'assertion qui y correspond à la liste des assertions représentant la base de faits. Les éléments du pattern sont des atomes, des listes-variable d'indicateur '<<' et '<+' ou des listes de ces mêmes éléments. L'assertion correspondant au pattern est obtenue en remplaçant chacune des listes-variable par la valeur associée à la variable dans une liste d'associations associée au moteur.

Si le conséquent est une opération, le moteur évalue les arguments de l'opération et leur applique la fonction.

Si le conséquent est un accès, le moteur évalue les éléments de la liste de valeurs, ajoute ou modifie les attributs de l'assertion identifiée.

9.1.7. Le "stream"

Le moteur d'inférence manipule une liste de listes d'associations lorsqu'il exécute les règles : cette structure de données est utilisée pour ranger les résultats de l'exécution d'une règle. Nous l'appelons un "stream". Nous présentons la manipulation des streams par le moteur d'inférence en 9.1.7.

Puisqu'un stream est une liste, nous pouvons utiliser des primitives de Mulisp pour manipuler les streams : car, cdr, cons, etc. Cependant, pour rendre les programmes plus clairs et plus simples à modifier, nous définissons des fonctions de manipulation des streams :

```
combine-streams
  arguments: s1 et s2
        s1 et s2 sont deux streams
```

résultat : s
la fonction renvoie le stream s qui la concaténation des streams s1 et s2

add-to-stream
arguments : e et s
e est un objet et s est un stream
résultat : s'
la fonction renvoie s', le stream construit en ajoutant e au début du stream s

first-of-stream
argument : s
s est un stream
résultat : e
la fonction renvoie e, le premier élément du stream s ; si le stream est vide, la fonction renvoie "nil"

rest-of-stream
argument : s
s est un stream
résultat : s'
la fonction renvoie s', le stream obtenu en retirant le premier élément du stream s

empty-stream
argument : s
s est un stream
résultat : "t" ou "nil"
la fonction renvoie "t" si le stream s est vide et "nil" sinon

make-empty-stream
argument :
résultat : nil
la fonction crée un stream vide, c'est-à-dire renvoie une liste vide ou "nil"

Le code de ces fonctions se trouve dans l'annexe.

9.1.8. Description du moteur d'inférence

9.1.8.1. Inférence sur une base de règles

Le moteur essaie d'appliquer les règles dans l'ordre dans lequel elles apparaissent dans la liste de règles. Si la règle courante réussit, le moteur a éventuellement ajouté des assertions à la liste d'assertions. Le moteur recommence alors à appliquer les règles à partir de la première de la liste de règles. Lorsqu'une règle échoue, le moteur essaie alors d'appliquer la règle suivante dans la liste.

Le moteur s'arrête lorsqu'il ne trouve plus de règle à appliquer, c'est-à-dire que la dernière règle de la liste de règles a échoué.

9.1.8.2. Exécution d'une règle

L'exécution comporte l'évaluation des antécédents et l'exécution des conséquents lorsque tous les antécédents réussissent.

L'évaluation d'un antécédent est la construction d'un stream résultat à partir d'un stream initial. Le stream initial contient des listes d'associations. Chacune contient les sous-listes (variable valeur) des associations réalisées grâce au "matching" lors de l'évaluation d'antécédents précédents.

Si l'antécédent est un pattern, le moteur considère le pattern et les listes d'associations l'une après l'autre. Lorsqu'il considère le pattern et une liste d'associations du stream initial, le moteur essaie de "matcher" le pattern avec les assertions de la liste d'assertions, l'une après l'autre. Les valeurs de variables sont puisées dans la liste d'associations. Les nouvelles sous-listes constituées sont déposées dans la liste d'associations. Pour chaque "matching" du pattern avec une assertion, la procédure de "pattern matching" produit une liste d'associations. Cette liste est la liste d'associations du stream initial complétée des sous-listes (variable valeur) évantuellement constituées par le "matching". La liste d'associations produite forme un objet du stream résultat. Pour un pattern et une liste d'associations du stream initial, le moteur produit autant de listes d'associations qu'il y a de "matching" entre le pattern et les assertions de la liste d'assertions. S'il n'y a pas de "matching", le moteur produit un stream résultat vide et l'assertion échoue.

Si l'antécédent est une opération-test, le moteur essaie d'appliquer la fonction spécifiée pour chacune des listes d'associations du stream initial et vérifie si le résultat renvoyé correspond au résultat attendu. Si c'est le cas, la liste d'associations est ajoutée au stream résultat. Si l'antécédent est une opération-calcul, le moteur applique la fonction aux listes d'associations du stream initial, et si l'opération réussit, ajoute à la liste d'associations la sous-liste constituée de la variable du résultat attendu et de la valeur renvoyée par la fonction. La liste d'associations ainsi complétée est ajoutée au stream résultat. Si l'opération ne réussit pour aucune des listes d'associations du stream initial, le stream résultat est vide. L'antécédent échoue.

Si l'antécédent échoue, le stream résultat est vide et le moteur n'évalue pas les antécédents suivants. Les conséquents ne sont pas exécutés. La règle échoue.

L'évaluation du premier antécédent est particulière. Le moteur n'a pas encore réalisé de "matching" ou exécuté d'opération pour cette règle. Dès lors, le stream initial ne contient pas de liste d'associations. Cependant, la règle n'a pas encore échoué. Donc, le stream initial n'est pas vide : il contient un élément, le symbole "nil". Le stream initial est donc s = (nil).

Conceptuellement, les antécédents peuvent être vus comme un ensemble de filtres placés en cascade. Leur évaluation est le filtrage d'un stream de listes d'associations au travers de la cascade qu'ils forment. Le résultat est un stream contenant autant de listes d'associations qu'il y a de "matching" sur les antécédents.

Le stream obtenu grâce au filtrage sur les antécédents doit ensuite être utilisé dans l'application des conséquents. Les conséquents sont des actions. Ils doivent être exécutés sur chacune des listes d'associations du stream. Le moteur considère ces listes l'une après l'autre. Il applique les actions l'une à la suite de l'autre sur une liste d'associations. Si le conséquent est un pattern, le moteur utilise la liste d'associations pour compléter le pattern, créer une nouvelle assertion et l'ajouter à la liste d'assertions représentant la base des faits et à un stream d'action. Si le conséquent est une opération, le moteur évalue les arguments de la fonction, éventuellement à partir de la liste d'associations lorsqu'il s'agit de variables à remplacer par leur valeur, applique la fonction. Si le conséquent est un accès, le moteur évalue les actions de l'accès, évantuellement à partir de la liste d'associations. Lorsque tous les conséquents ont été

appliqués à toutes les listes d'associations, si le stream d'action n'est pas vide, le moteur peut en déduire que la règle a réussi.

Conceptuellement, l'exécution des conséquents peut être vue comme la distribution d'un stream de listes d'associations entre différentes boîtes d'action que le moteur combine pour former un stream d'action.

9.1.8.3. Utilisation du moteur d'inférence

Dans l'architecture du système, nous avons deux bases de règles : la base des règles de connaissances et la base des méta-règles. Le moteur réalise une inférence, telle qu'elle est décrite en [9.1.7.1.], sur la base des méta-règles. L'exécution d'une méta-règle permet de déterminer les règles de connaissances qui doivent être appliquées. Ces règles sont exécutées comme nous l'avons décrit en [9.1.7.2.].

9.1.8.4. Fonctions du moteur d'inférence

Nous spécifions à présent les différentes fonctions du moteur d'inférence. Le code de ces fonctions peut être consulté dans l'annexe A.

filter-lstasso
  arguments : filter, lstasso et asserts
      filter est un filtre. Un filtre est un pattern ou une opération ; lstasso est une liste d'associations et asserts est la liste d'assertions représentant la base de faits
  résultat : s

effets de bord : effets de bord de filter ou aucun

filter-lstasso passe la liste d'associations lstasso dans le filtre filter et renvoie un stream s. Si filter est une opération, filter-lstasso essaie d'appliquer la fonction spécifiée dans l'opération en s'aidant éventuellement de la liste d'associations lstasso pour déterminer les arguments de la fonction. Les effets de bord sont ceux de l'application de la fonction spécifiée dans l'opération filter. Si l'opération réussit, filter-lstasso renvoie s contenant une liste d'associations. Cette liste est lstasso complétée de la sous-liste constituée d'une variable et du résultat renvoyé par la fonction lorsque l'opération est une opération-calcul. Cette liste est lstasso lorsque l'opération est une opération-test. Si l'opération ne réussit pas, filter-lstasso renvoie s vide. Si le filtre est un pattern, filter-lstasso essaie de "matcher" le pattern avec toutes les assertions d'asserts, en s'aidant éventuellement de la liste d'associations lstasso pour remplacer les variables précédées des indicateurs de variable '<<' et '<+'. A chaque "matching", une liste d'associations est produite, composée de lstasso et des associations réalisées par le "matching". La liste d'associations produite est ajoutée au stream s qui est renvoyé par filter-lstasso lorsque toute la liste d'assertions asserts a été parcourue. Si aucun "matching" ne se produit, filter-lstasso renvoie s vide. Si filter est un pattern, il n'y a aucun effet de bord.

filter-stream

arguments : filter et s

filter est un filtre et s est un stream

résultat : s'

effets de bord : effets de bord de filter ou aucun

filter-stream passe un stream s dans un filtre filter. Les listes d'associations du stream s sont passées l'une après l'autre dans le filtre filter de la manière spécifiée dans filter-lstasso. Pour chacun de ces filtrages est produit un stream. La fonction renvoie s', un stream obtenu par la concaténation des streams produits pour chaque filtrage de liste d'associations de s. Si filter est une opération, l'application de la fonction de l'opération à chacune des listes d'associations peut produire des effets de bord. Si filter est un pattern, il n'y a aucun effet de bord

cascade-through-filters

arguments : filters et s

filters est une liste de filtres et s est un stream

résultat : s'

effets de bord : effets de bord des filtres ou aucun

la fonction passe un stream s au travers d'une cascade de filtres filters. Le passage d'un stream dans un filtre produit un stream de la manière spécifiée dans filter-stream. Le stream obtenu en résultat du filtrage courant est passé dans le filtre suivant de filters. Le stream s est passé dans le premier filtre. Le stream obtenu après le passage dans le dernier filtre est s' qui est renvoyé par la fonction. Si un filtre de filters est une opération, le passage du stream dans ce filtre peut produire des effets de bord. Les effets de bord de cascade-through-filters est l'ensemble des effets de bord produit par les différentes opérations qui constituent filters. Si un filtre de filters est un pattern, aucun effet de bord n'est produit.

execute-action
 arguments : action, lstasso et asserts
 Une action est un pattern, une opération ou un accès ; lstasso est une liste d'associations et asserts est la liste d'assertions représentant la base de faits
 résultat : lst
 effets de bord : asserts
 effets de bord des opérations
 la fonction exécute une action sur une liste d'associations lstasso. Si l'action est un pattern, une assertion lst est créée à partir du pattern en utilisant la liste d'associations lstasso pour remplacer les variables du pattern par leur valeur. L'assertion est ajoutée à la liste d'assertions asserts et execute-action renvoie lst. Si l'action est une opération, la fonction indiquée est appliquée et execute-action renvoie lst = (ope). Si l'action est un accès, execute-action réalise l'accès à l'assertions de asserts identifiée par l'action, réalise les ajouts et les modifications spécifiées dans l'accès et renvoie lst = (id).

spread-through-actions
 arguments : actions, lstasso et asserts
 actions est une liste d'actions ; lstasso est une liste d'associations et asserts est la liste d'assertions représentant la base de faits
 résultat : s
 effets de bord : asserts
 effets de bord des opérations
 la fonction exécute les actions l'une après l'autre sur une liste d'associations lstasso de la manière déterminée dans execute-action. La fonction spread-through-actions renvoie un stream d'action s construit à partir de toutes les actions exécutées :

un pattern entraîne l'ajout de l'assertion créée au stream d'action s, une opération entraîne l'ajout de l'expression (ope) au stream d'action s et un accès l'ajout de l'expression (id).

feed-to-actions

arguments : actions et s

actions est une liste d'actions et s est stream

résultat : s'

effets de bord : effets de bord des opérations ou aucun

la fonction exécute un ensemble d'actions actions sur un stream s. Elle exécute l'ensemble d'actions sur chacune des listes d'associations du stream s à tour de rôle de la manière spécifiée dans spread-through-actions. Chacune de ces exécutions renvoie un stream d'action. La fonction renvoie un stream d'action s' qui est la concaténation de tous les streams d'action renvoyés pour les différentes exécutions. Si une action est opération, elle peut éventuellement avoir des effets de bord.

use-rule

arguments : rule

rule est une règle

résultat : "t" ou "nil"

la fonction applique une règle. Elle extrait les antécédents et les conséquents de la règle. Les antécédents forment une cascade de filtres et les conséquents forment un suite d'actions. La cascade de filtres est appliquée et produit un stream. La liste d'actions est appliquée à ce stream s'il n'est pas vide. Si la liste d'actions peut être exécutée et produire un stream d'action non vide, la fonction renvoie "t", sinon elle renvoie "nil".

forward-chaining
  argument : rules-to-try
  résultat : "t" ou "nil"
      forward-chaining exploite la liste des règles rules-to-try comme il a été défini en 9.1.7.1.

9.2. IMPLEMENTATION DE LA BASE DE CONNAISSANCES

9.2.1. Spécification des primitives utilisées dans les règles

Fonctions de manipulation de listes

compare
  arguments : ope, elem et lst
      ope est un identificateur de fonction de comparaison de nombres, elem est un nombre et lst est une liste de nombres (l1 l2 ... ln)
  résultat : lst'
      compare renvoie une liste lst' composés de tous les éléments li de lst tels que l'application de la fonction ope à elem et li (ope elem li) renvoie "t" ; si aucun élément li de lst ne satisfait cette condition, compare renvoie une liste vide

succession
  arguments : l1, l2 et l3
      l1, l2 et l3 sont des listes de nombres :
      l1 = (l11 l12 ... l1n)
      l2 = (l21 l22 ... l2n)
      l3 = (l31 l32 ... l3n)
  résultat : lst
      succession renvoie une liste lst composés de tous les éléments l1i de l1 tels que, pour l1i existe l2j appartenant à l2 : (l2j - 1) = l1i, et existe l3k

appartenant à l3 : (l3k + 1) = l1i ; si aucun élément de l1 ne satisfait ces conditions, succession renvoie une liste vide

add-1
  argument : lst
    lst est une liste de nombres
  résultat : lst'
    add-1 renvoie lst', une liste obtenue en ajoutant 1 aux éléments de lst

sub-1
  argument : lst
    lst est une liste de nombres
  résultat : lst'
    sub-1 renvoie lst', une liste obtenue en retirant 1 aux éléments de lst

nearest
  argument : dir nbr lstasso
    dir est l'atome "below" ou l'atome "above" ; nbr est un nombre ; lstasso est une liste d'associations dont les clés sont k1, k2, ..., kn et les valeurs de ces clés sont des listes croissantes de nombres
  résultat : lstkey
    si dir est égal à "below", nearest renvoie lstkey, une liste (c1 c2 ... cr) telle que pour tout cj, 1 <= j <= r, existe ki, 1 <= i <= n, et ki = cj, et les valeurs de ces clés dans lstasso sont (v11 v12 ... v1m), (v21 v22 ... v2o) ... (vr1 vr2 ... vrp), et (v11 - 1- nbr) = (v21 - 1 - nbr) = ... = (vr1 - 1 - nbr), et il n'existe aucun kh tel que (vh1 - 1 - nbr) < (v11 - 1 - nbr), 1 <= h <= n et 1 <= l <= r
    si dir est égal à "above", nearest renvoie lstkey, une liste (c1 c2 ... cr) telle que pour tout cj,

1 <= j <= r, existe ki, 1 <= i <= n, et ki = cj, et les valeurs de ces clés dans lstasso sont (v11 v12 ... v1m), (v21 v22 ... v2o) ... (vr1 vr2 ... vrp), et (nbr - 1- v1m) = (nbr - 1 - v2o) = ... = (nbr - 1 - vrp), et il n'existe aucun kh tel que (nbr - 1 - vhs) < (nbr - 1 - vlq), 1 <= h <= n et 1 <= l <= r, vhs et vlq étant les derniers éléments de la liste de nombres auquels ils appartiennent

Fonctions de calcul

diameter
  arguments : vol et heig
      vol et heig sont deux nombres
  résultat : diam
      diameter renvoie diam, le diamètre du cylindre dont le volume est vol et la hauteur heig

equation
  argument : x0, x1, y0, y1, x
      x0, x1, y0, y1 et x sont des nombres ; (x0,x1) et (y0,y1) sont deux points d'une droite d ; x est l'abscisse d'un point de cette même droite d
  résultat : y
      equation renvoie y, un nombre tel que le point (x,y) appartient à la droite d

Fonctions d'interface

print-tech
  argument : technique
      technique est une liste d'associations dont les sous-listes ont pour clé les caractéristiques de

l'agrégat technique :
((nom v1) (d v2) (h v3) (ph v4) (gel v5) (volgel v6)
(tampon v7) (t v8) (elution v9) (perte v10)
(volagprec v11)
), avec v1 à v11 les valeurs des différentes clés
effet de bord : affichage des caractéristiques à l'écran
print-tech "nettoie" l'écran et affiche les caractéristiques de la technique dont la valeur est différente de "nil"

print-sample
argument : sample
sample est une liste d'associations dont les sous-listes ont pour clé les caractéristiques de l'agrégat échantillon :
((melange v1) (vol v2) (q v3) (cc v4) (proteine v5)
(contaminants v6)
), avec v1 à v6 les valeurs des différentes clés
résultat : "t" ou "nil"
effet de bord : affichage des caractéristiques à l'écran
print-sample affiche les caractéristiques de l'échantillon dont la valeur est différente de "nil" et demande à l'utilisateur s'il accepte l'application de la technique permettant d'obtenir les caractéristiques de sample ; si l'utilisateur répond affirmativement, print-sample renvoie "t" ; si l'utilisateur répond négativement, print-sample renvoie "nil"

question-height
résultat : height
question-height renvoie height, une hauteur de colonne demandée à l'utilisateur comprise entre 10 et 20cm

question-decroch
  résultat : decroch
      question-decroch renvoie "e" ou "g", signifiant si le décrochage lors d'un échangeur d'ions doit se faire par étapes ou par gradient. Cette information est demandée à l'utilisateur

9.2.2. Règles concernant la chromatographie sur échangeur d'ions

Nous présentons ici, en langage naturel, les règles de la base de connaissances. Le lecteur intéressé trouvera le code de ces règles en annexe.

9.2.2.1. Choix d'application de la technique

- Nom de la règle : ei1.

- Antécédents : les antécédents de cette règle "matchent" dans la mémoire de travail les substances contaminantes de l'échantillon, leur point isoélectrique, la protéine à purifier et son point isoélectrique. Ils ne gardent ensuite que les contaminants dont le point isoélectrique n'est pas compris dans l'intervalle [point isoélectrique de la protéine à purifier - 1, point isoélectrique de la protéine à purifier + 1] ou est égal a une des bornes de cet intervalle.

- Conséquent : le conséquent consiste à créer dans la mémoire de travail une assertion qui est une liste dont le premier élément est l'atome identifiant "éliminés" et dont l'élément suivant est une liste des contaminants pour lesquels les antécédents ont réussi. Il s'agit des substances contaminantes qui seront éliminées par l'application éventuelle de la technique de l'échangeur

d'ions.

- Nom de la règle : ei2.

- Antécédents : les antécédents de cette règle "matchent" dans la mémoire de travail les substances contaminantes de l'échantillon, leur point isoélectrique, la protéine à purifier et son point isoélectrique. Ils ne gardent ensuite que les contaminants dont le point isoélectrique est compris dans l'intervalle [point isoélectrique de la protéine à purifier - 1, point isoélectrique de la protéine à purifier + 1].

- Conséquent : le conséquent consiste à créer dans la mémoire de travail une assertion qui est une liste dont le premier élément est l'atome identifiant "contaminants" et l'élément suivant est une liste des contaminants pour lesquels les antécédents ont réussi. Il s'agit des substances contaminantes qui ne seront pas éliminées par l'application éventuelle de la technique de l'échangeur d'ions.

- Nom de la règle : ei3.

- Antécédents : les antécédents de cette règle "matchent" dans la mémoire de travail la liste des substances éliminées créée par la règle ei1 et la liste des substances contaminantes créée par la règle ei2. Ils vérifient ensuite que le nombre de substances éliminées est plus grand ou égal au nombre de substances contaminantes

- Conséquents : les conséquents créent dans la mémoire de

travail une assertion "technique" et sa caractéristique "nom" qui prend la valeur "ei". Ils enlèvent de la mémoire de travail les deux assertions créées par les règles ei1 et ei2.

Ces trois règles assurent que la technique de l'échangeur d'ions ne sera appliquée que si le nombre de substances contaminantes que l'on pourra éliminer est supérieur ou égal au nombre de substances qui continueront à contaminer la protéine à purifier, après application de la technique.

9.2.2.2. Evaluation de l'application de la technique

- Nom de la règle : ei4.

- Antécédents : les antécédents de cette règle "matchent" dans la mémoire de travail la protéine à purifier, son point isoélectrique et son domaine de stabilité. Ils assurent que le point isoélectrique est plus éloigné de la borne inférieure du domaine de stabilité que de la borne supérieure.

- Conséquent : le conséquent ajoute à l'assertion "technique" sa caractéristique "ph" qui prend la valeur de la borne inférieure du domaine de stabilité.

- Nom de la règle : ei5.

- Antécédents : les antécédents de cette règle "matchent" dans la mémoire de travail la protéine à purifier, son point isoélectrique et son domaine de stabilité. Ils assurent que le point isoélectrique est plus éloigné de la borne

supérieure du domaine de stabilité que de la borne inférieure.

- Conséquent : le conséquent ajoute à l'assertion "technique" sa caractéristique "ph" qui prend la valeur de la borne supérieure du domaine de stabilité.

Ces deux règles déterminent le pH auquel l'opérateur doit effectuer la purification. Elles assurent que ce pH est le plus éloigné du point isoélectrique de la protéine à purifier, tout en restant dans le domaine de stabilité de cette protéine.
- Nom de la règle : ei6.

- Antécédents : les antécédents de cette règle "matchent" dans la mémoire de travail le pH de purification conseillé pour la technique, la protéine à purifier et son point isoélectrique. Ils vérifient que le pH est supérieur au point isoélectrique.

- Conséquent : le conséquent ajoute à l'assertion "technique" sa caractéristique "gel" qui prend la valeur "DEAE Sephadex-A50".

- Nom de la règle : ei7.

- Antécédents : les antécédents de cette règle "matchent" dans la mémoire de travail le pH de purification conseillé pour la technique, la protéine à purifier et son point isoélectrique. Ils vérifient que le pH est inférieur au point isoélectrique et que le pH est inférieur ou égal à 5.

- Conséquent : le conséquent ajoute à l'assertion "technique" sa caractéristique "gel" qui prend la valeur "SP Sephadex-C50".

- Nom de la règle : ei8.

- Antécédents : les antécédents de cette règle "matchent" dans la mémoire de travail le pH de purification conseillé pour la technique, la protéine à purifier et son point isoélectrique. Ils vérifient que le pH est inférieur au point isoélectrique et que le pH est supérieur à 5.

- Conséquent : le conséquent ajoute à l'assertion "technique" sa caractéristique "gel" qui prend la valeur "CM Sephadex-C50".

Ces trois règles déterminent le type de gel à utiliser pour la purification.

- Nom de la règle : ei9.

- Antécédents : les antécédents de cette règle vérifient qu'un type de gel a bien été déterminé et "matchent" dans la mémoire de travail la quantité de protéines dans l'échantillon. Ils calculent un volume en fonction de cette quantité.

- Conséquent : le conséquent ajoute à l'assertion "technique" sa caractéristique "volgel" qui prend la valeur du volume calculé dans les antécédents.

Cette règle détermine le volume de gel à utiliser pour l'application de la technique de purification.

- Nom de la règle : ei10.

- Antécédents : les antécédents de cette règle vérifient qu'un volume de gel a bien été déterminé et "matchent" dans la mémoire de travail les caractéristiques de l'assertion "technique" déjà construite. Ils présentent à l'utilisateur cette assertion, à l'aide de la primitive print-tech, et ils lui demandent la hauteur de la colonne qu'il compte utiliser, par la primitive question-height. Ils appliquent ensuite la primitive diameter avec les arguments hauteur et vol pour calculer le diamètre de la colonne en fonction de sa hauteur et du volume de gel qu'elle aura à contenir.

- Conséquent : le conséquent ajoute à l'assertion "technique" sa caractéristique "h" qui prend la valeur de la hauteur déterminée par l'utilisateur dans les antécédents et sa caractéristique "d" qui prend la valeur du diamètre calculé dans les antécédents.

- Nom de la règle : ei11.

- Antécédents : les antécédents de cette règle vérifient qu'un diamètre de colonne a bien été déterminé et "matchent" dans la mémoire de travail le pH de l'assertion "technique" déjà construite, la protéine à purifier et son point isoélectrique, les tampons et leur(s) pk. Ils assurent que le point isoélectrique de la protéine et le pH de purification sont compris dans un intervalle [pk-1, pk+1]

pour chaque tampon, en utilisant la primitive compare.

- Conséquent : le conséquent ajoute à l'assertion "technique" sa caractéristique "tampon" qui prend pour valeur la liste de noms de tampons qui ont satisfait les antécédents.

- Nom de la règle : ei12.

- Antécédents : les antécédents de cette règle vérifient qu'un diamètre de colonne a bien été déterminé et "matchent" dans la mémoire de travail le pH de l'assertion "technique" déjà construite, la protéine à purifier et son point isoélectrique, les tampons et leur(s) pk. Ils assurent que le pH de purification est compris dans un intervalle [pk-1, pk+1] et que le point isoélectrique de la protéine est inférieur à la borne inférieure de cet intervalle, pour chaque tampon. Ils utilisent la primitive compare. Il reste alors, parmi les tampons qui vérifient ces conditions, à construire une liste des pk associés à ces tampons qui sont le centre des intervalles [pk-1, pk+1] qui ont satisfait les conditions ci-dessus. La primitive succession est utilisée.

- Conséquent : le conséquent crée dans la mémoire de travail une assertion dont le premier élément est l'atome identifiant "tampon-pi-inf" et dont le second élément est une liste des tampons et de leur(s) pk qui ont satisfait les antécédents.

- Nom de la règle : ei13.

- Antécédents : les antécédents "matchent" dans la mémoire de travail la protéine à purifier et son point isoélectrique, la liste de tampons et leur(s) pk de l'assertion créée par la règle précédente. Ils utilisent la primitive nearest pour ne garder que le(s) tampon(s) dont la borne inférieure de l'intervalle [pk-1, pk+1] est la plus proche du point isoélectrique de la protéine à purifier.

- Conséquent : le conséquent ajoute à l'assertion "technique" sa caractéristique "tampon" qui prend pour valeur la liste de noms de tampons qui ont satisfait les antécédents.

- Nom de la règle : ei14.

- Antécédents : les antécédents de cette règle vérifient qu'un diamètre de colonne a bien été déterminé et "matchent" dans la mémoire de travail le pH de l'assertion "technique" déjà construite, la protéine à purifier et son point isoélectrique, les tampons et leur(s) pk. Ils assurent que le pH de purification est compris dans un intervalle [pk-1, pk+1] et que le point isoélectrique de la protéine est supérieur à la borne supérieure de cet intervalle, pour chaque tampon. Ils utilisent la primitive compare. Il reste alors, parmi les tampons qui vérifient ces conditions, à construire une liste des pk associés à ces tampons qui sont le centre des intervalles [pk-1, pk+1] qui ont satisfait les conditions ci-dessus. La primitive succession est utilisée.

- Conséquent : le conséquent crée dans la mémoire de travail une assertion dont le premier élément est l'atome

identifiant "tampon-pi-sup" et dont le second élément est une liste des tampons et de leur(s) pk qui ont satisfait les antécédents.

- Nom de la règle : ei15.

- Antécédents : les antécédents "matchent" dans la mémoire de travail la protéine à purifier et son point isoélectrique, la liste de tampons et leur(s) pk de l'assertion créée par la règle précédente. Ils utilisent la primitive nearest pour ne garder que le(s) tampon(s) dont la borne supérieure de l'intervalle [pk-1, pk+1] est la plus proche du point isoélectrique de la protéine à purifier.

- Conséquent : le conséquent ajoute à l'assertion "technique" sa caractéristique "tampon" qui prend pour valeur la liste de noms de tampons qui ont satisfait les antécédents.

Les six règles ci-dessus proposent le(s) tampon(s) utilisable(s) en fonction du point isoélectrique de la protéine à purifier et du pH d'application de la technique.

- Nom de la règle : ei16.

- Antécédents : les antécédents vérifient qu'un ou plusieurs tampons ont bien été déterminés. Ils "matchent" dans la mémoire de travail la protéine à purifier, son point isoélectrique et son domaine de stabilité. Ils assurent que le point isoélectrique est supérieur à la borne supérieure du domaine de stabilité.

- Conséquent : le conséquent ajoute à l'assertion "technique" sa caractéristique "elution" qui prend la valeur "force ionique".

- Nom de la règle : ei17.

- Antécédents : les antécédents vérifient qu'un ou plusieurs tampons ont bien été déterminés. Ils "matchent" dans la mémoire de travail la protéine à purifier, son point isoélectrique et son domaine de stabilité. Ils assurent que le point isoélectrique est inférieur à la borne inférieure du domaine de stabilité.

- Conséquent : le conséquent ajoute à l'assertion "technique" sa caractéristique "elution" qui prend la valeur "force ionique".

- Nom de la règle : ei18.

- Antécédents : les antécédents vérifient qu'un ou plusieurs tampons ont bien été déterminés. Ils "matchent" dans la mémoire de travail la protéine à purifier, son point isoélectrique et son domaine de stabilité. Ils assurent que le point isoélectrique est compris dans le domaine de stabilité.

- Conséquent : le conséquent ajoute à l'assertion "technique" sa caractéristique "elution" qui prend la valeur "force ionique et pH".

Ces trois règles déterminent le type d'élution à utiliser suivant que le point isoélectrique de la protéine à purifier se trouve ou non dans son domaine de stabilité.

- Nom de la règle : ei19.

- Antécédents : les antécédents vérifient que le type d'élution a bien été déterminé et "matchent" dans la mémoire de travail les caractéristiques de l'assertion "technique" déjà construite. Ils présentent à l'utilisateur cette assertion, à l'aide de la primitive print-tech, et ils lui demandent le type de décrochage qu'il compte appliquer, par la primitive question-decroch.

- Conséquent : le conséquent ajoute à l'assertion "technique" sa caractéristique "decroch" qui prend la valeur donnée par l'utilisateur : par gradient ou en étapes.

- Nom de la règle : ei20 et ei21.

- Antécédents : les antécédents "matchent" le volume de gel et le type de décrochage conseillés pour l'application de la technique. Ils calculent ensuite le volume de l'échantillon résultant de l'application de la technique en fonction du volume de gel et du type de décrochage.

- Conséquent : le conséquent modifie la caractéristique "vol" de l'assertion "echantillon", en lui donnant la valeur calculée dans les antécédents.

- Nom de la règle : ei22.

- Antécédents : les antécédents "matchent" les substances contaminantes de l'échantillon, leur point isoélectrique et leur proportion.

- Conséquents : les conséquents modifient la caractéristique "contaminants" de l'assertion "echantillon", en lui donnant la valeur "nil". Ils créent dans la mémoire de travail une assertion dont le premier élément est l'atome identifiant "contaminant" et dont les éléments suivants sont le nom du contaminant, son point isoélectrique et sa proportion. Une telle assertion est créée pour chaque contaminant qui a satisfait les antécédents.

- Nom de la règle : ei23 et ei24.

- Antécédents : les antécédents "matchent" l'assertion "contaminant" créée par la règle précédente, la protéine à purifier et son point isoélectrique. Ils vérifient que le point isoélectrique des contaminants est compris dans l'intervalle [point isoélectrique de la protéine à purifier - 1, point isoélectrique de la protéine à purifier + 1]. Ils calculent pour les contaminants vérifiant ces conditions le pourcentage de contamination après application de la technique, en utilisant la primitive equation.

- Conséquents : les conséquents retirent de la mémoire de travail, pour les contaminants qui vérifient les antécédents, l'assertion créée par la règle précédente. Ils créent pour chacun de ces mêmes contaminants une assertion dont le premier élément est l'identifiant "element de contamination" et dont les éléments suivants sont le nom du contaminant et sa proportion.

- Nom de la règle : ei25.

- Antécédents : les antécédents "matchent" l'assertion "contaminant" créée par la règle ei22, la protéine à purifier et son point isoélectrique. Ils vérifient que le point isoélectrique des contaminants n'est pas compris dans l'intervalle [point isoélectrique de la protéine à purifier - 1, point isoélectrique de la protéine à purifier + 1].

- Conséquents : les conséquents retirent de la mémoire de travail, pour les contaminants qui vérifient les antécédents, l'assertion créée par la règle ei22.

- Nom de la règle : ei26.

- Antécédents : les antécédents "matchent" l'assertion "element de contamination". Ils vérifient que la proportion des contaminants est différente de 0.

- Conséquents : les conséquents retirent de la mémoire de travail, pour les contaminants qui vérifient les antécédents, l'assertion "element de contamination". Ils modifient l'assertion "echantillon", en ajoutant à la caractéristique "contaminants" la liste des contaminants qui ont vérifié les antécédents et leur proportion.

Ces six règles détermiment les substances qui contaminent l'échantillon résultat, après application de la technique.

9.2.3. Règles concernant la précipitation isoélectrique

Nous présentons ici, en langage naturel, les règles de la base de connaissances. Le lecteur intéressé trouvera le code de ces règles en annexe.

9.2.3.1. Choix d'application de la technique

- Nom de la règle : pr1.

- Antécédents : les antécédents "matchent" dans la mémoire de travail la protéine à purifier et son domaine de précipitation. Ils vérifient que la borne supérieure de ce domaine est inférieure à 60.

- Conséquent : le conséquent crée dans la mémoire de travail une assertion "technique" et sa caractéristique "nom" qui prend la valeur "pr".

Cette règle assure que la technique de la précipitation isoélectrique ne sera appliquée que si la protéine à purifier précipite avant 60%.

9.2.3.2. Evaluation de l'application de la technique

- Nom de la règle : pr2.

- Antécédents : les antécédents "matchent" dans la mémoire de travail le volume de l'échantillon, la protéine à purifier et son domaine de précipitation. Ils calculent le volume de l'agent de précipitation à partir de ces données.

- Conséquent : le conséquent ajoute à l'assertion "technique" sa caractéristique "volagprec" qui prend la valeur calculée dans les antécédents.

- Nom de la règle : pr3.

- Antécédents : les antécédents "matchent" les substances contaminantes de l'échantillon, leur domaine de précipitation et leur proportion.

- Conséquents : les conséquents modifient la caractéristique "contaminants" de l'assertion "echantillon", en lui donnant la valeur "nil". Ils créent dans la mémoire de travail une assertion dont le premier élément est l'atome identifiant "contaminant" et dont les éléments suivants sont le nom du contaminant, son domaine de précipitation et sa proportion. Une telle assertion est créée pour chaque contaminant qui a satisfait les antécédents.

- Nom de la règle : pr4.

- Antécédents : les antécédents "matchent" l'assertion "contaminant" créée par la règle précédente, la protéine à purifier et son domaine de précipitation. Ils vérifient que la borne supérieure du domaine de précipitation de la protéine à purifier est inférieure ou égale à la borne inférieure du domaine de précipitation des contaminants.

- Conséquents : les conséquents retirent de la mémoire de travail, pour les contaminants qui vérifient les antécédents, l'assertion créée par la règle précédente. Ils modifient l'assertion "echantillon", en ajoutant à la caractéristique "contaminants" la liste des contaminants qui ont vérifié les antécédents et leur proportion.

- Nom de la règle : pr5.

- Antécédents : les antécédents "matchent" l'assertion "contaminant" créée par la règle pr3. Cette assertion ne "matche" que pour les contaminants dont le domaine de précipitation contient la borne supérieure du domaine de précipitation de la protéine à purifier, les autres ayant été retirés de la mémoire de travail par les règles précédentes. Ils "matchent" également la protéine à purifier et son domaine de précipitation. Ils calculent à, partir de ces informations, le pourcentage de contamination des contaminants en utilisant la primitive equation.

- Conséquents : les conséquents retirent de la mémoire de travail, pour les contaminants qui vérifient les antécédents, l'assertion créée par la règle pr3.

- Nom de la règle : pr6.

- Antécédents : les antécédents "matchent" l'assertion "contaminant" créée par la règle précédente, la protéine à purifier et son domaine de précipitation. Ils vérifient que la borne supérieure du domaine de précipitation de la protéine à purifier est supérieure ou égale à la borne supérieure du domaine de précipitation des contaminants.

- Conséquents : les conséquents retirent de la mémoire de travail, pour les contaminants qui vérifient les antécédents, l'assertion créée par la règle pr3. Ils modifient l'assertion "echantillon", en ajoutant à la caractéristique "contaminants" la liste des contaminants qui ont vérifié les antécédents et leur nouvelle proportion.

Ces quatre règles détermiment les substances qui contaminent l'échantillon résultat, après application de la technique.

9.2.4. Règles concernant la chromatographie sur tamis moléculaire

9.2.4.1. Choix d'application de la technique

- Nom de la règle : tm1.

- Antécédents : les antécédents "matchent" la protéine à purifier, sa proportion et vérifient que cette proportion est inférieure à 10%.

- Conséquent : le conséquent crée une assertion "volume envisageable".

- Nom de la règle : tm2.

- Antécédents : les antécédents vérifient qu'une assertion "volume envisageable" a été crée. Ils "matchent" le volume de l'échantillon et vérifient que ce volume est supérieur à 20ml. Ils "matchent" la quantité de substances dans l'échantillon et vérifient que cette quantité est supérieure à 10mg/ml.

- Conséquents : les conséquents, par une primitive d'interface, conseillent à l'utilisateur de ne pas appliquer un tamis moléculaire. Cependant, si l'utilisateur veut malgré tout l'appliquer, ils lui demandent son choix entre les possibilités présentées dans la première situation du

point [7.3.1.2.2.] Ils créent une assertion "poids moléculaire envisageable".

- Nom de la règle : tm3.

- Antécédents : les antécédents vérifient qu'une assertion "volume envisageable" a été crée. Ils "matchent" le volume de l'échantillon et vérifient que ce volume est supérieur à 20ml. Ils "matchent" la quantité de substances dans l'échantillon et vérifient que cette quantité est égale à 10mg/ml.

- Conséquents : les conséquents, par une primitive d'interface, signalent à l'utilisateur qu'un tamis moléculaire est envisageable. Ils lui demandent son choix entre les possibilités présentées dans la deuxième situation du point [7.3.1.2.2.] Ils créent une assertion "poids moléculaire envisageable".

- Nom de la règle : tm4.

- Antécédents : les antécédents vérifient qu'une assertion "volume envisageable" a été crée. Ils "matchent" le volume de l'échantillon et vérifient que ce volume est supérieur à 20ml. Ils "matchent" la quantité de substances dans l'échantillon et vérifient que cette quantité est inférieure à 10mg/ml.

- Conséquents : les conséquents, par une primitive d'interface, demandent à l'utilisateur son choix entre les possibilités présentées dans la troisième situation du point

[7.3.1.2.2.] Ils créent une assertion "poids moléculaire envisageable".

- Nom de la règle : tm5.

- Antécédents : les antécédents vérifient qu'une assertion "volume envisageable" a été crée. Ils "matchent" le volume de l'échantillon et vérifient que ce volume est inférieur à 20ml. Ils "matchent" la quantité de substances dans l'échantillon et vérifient que cette quantité est supérieure à 10mg/ml.

- Conséquents : les conséquents, par une primitive d'interface, demandent à l'utilisateur son choix entre les possibilités présentées dans la quatrième situation du point [7.3.1.2.2.] Ils créent une assertion "poids moléculaire envisageable".

- Nom de la règle : tm6.

- Antécédents : les antécédents vérifient qu'une assertion "volume envisageable" a été crée. Ils "matchent" le volume de l'échantillon et vérifient que ce volume est inférieur à 20ml. Ils "matchent" la quantité de substances dans l'échantillon et vérifient que cette quantité est égale à 10mg/ml.

- Conséquents : les conséquents, par une primitive d'interface, proposent à l'utilisateur d'appliquer un tamis moléculaire. Ils créent une assertion "poids moléculaire envisageable".

- Nom de la règle : tm7.

- Antécédents : les antécédents vérifient qu'une assertion "volume envisageable" a été crée. Ils "matchent" le volume de l'échantillon et vérifient que ce volume est inférieur à 20ml. Ils "matchent" la quantité de substances dans l'échantillon et vérifient que cette quantité est inférieure à 10mg/ml.

- Conséquents : les conséquents, par une primitive d'interface, demandent à l'utilisateur son choix entre les possibilités présentées dans la sixième situation du point [7.3.1.2.2.] Ils créent une assertion "poids moléculaire envisageable".

Ces six règles assurent que le volume et la quantité de protéine dans l'échantillon permettent l'application d'un tamis moléculaire.

- Nom de la règle : tm8.

- Antécédents : les antécédents vérifient qu'une assertion "poids moléculaire envisageable" a été crée. Ils "matchent" le volume de l'échantillon et les assertions décrivant des colonnes. Ils ne gardent que la plus petite colonne qui accepte le volume de l'échantillon.

- Conséquents : les conséquents créent dans la mémoire de travail une assertion "technique" et sa caractéristique "nom" qui prend la valeur "tm", sa carctéristique "h" qui prend la valeur de la hauteur de la colonne déterminée par les antécédents, sa caractéristique "d" qui prend la valeur du diamètre de la colonne déterminée dans les antécédents et

sa caractéristique "volcol" qui prend la valeur du volume maximum possible pour la colonne déterminée par les antécédents.

- Nom de la règle : tm9.

- Antécédents : les antécédents vérifient qu'une assertion "technique" a été crée ainsi que les caractéristiques déterminées par la règle précédente. Ils "matchent" les assertions "gel", la protéine à purifier et son poids moléculaire. Ils calculent le domaine de non-séparation de la protéine et gardent les gels dont le domaine de fractionnement contient le domaine de non-séparation.

- Conséquent : le conséquent crée dans la mémoire de travail une assertion dont le premier élément est l'atome identifiant "gel-dom-fr" et dont le second élément est une liste des gels et de leur domaine de non séparation qui ont satisfait les antécédents.

- Nom de la règle : tm10.

- Antécédents : les antécédents de cette règle "matchent" dans la mémoire de travail les substances contaminantes de l'échantillon, leur poids moléculaire, la protéine à purifier, son poids moléculaire et l'assertion "gel-dom-fr" créée par la règle précédente.

- Conséquent : le conséquent consiste à créer dans la mémoire de travail, pour chaque gel, une assertion qui est une liste dont le premier élément est l'atome identifiant "éliminés" et l'élément suivant est une liste des

contaminants dont le poids moléculaire n'est pas compris dans le domaine de non-séparation du gel. Il s'agit des substances contaminantes qui seront éliminées par l'application éventuelle de la technique du tamis moléculaire avec ce gel.

- Nom de la règle : tm11.

- Antécédents : les antécédents de cette règle "matchent" dans la mémoire de travail les substances contaminantes de l'échantillon, leur poids moléculaire, la protéine à purifier, son poids moléculaire et l'assertion "gel-dom-fr" créée par la règle précédente.

- Conséquent : le conséquent consiste à créer dans la mémoire de travail, pour chaque gel, une assertion qui est une liste dont le premier élément est l'atome identifiant "contaminants" et l'élément suivant est une liste des contaminants dont le poids moléculaire est compris dans le domaine de non-séparation du gel. Il s'agit des substances contaminantes qui ne seront pas éliminées par l'application éventuelle de la technique du tamis moléculaire avec ce gel.

- Nom de la règle : tm12.

- Antécédents : les antécédents de cette règle "matchent" dans la mémoire de travail les assertions créées par les deux règles précédentes. Ils ne gardent que l'assertion "elimines" correspondant au gel qui permet d'éliminer le plus grand nombre de substances. Pour ce gel, ils vérifient que le nombre de substances éliminées est plus grand que le nombre de substances contaminantes.

- Conséquent : le conséquent modifie la caractéristique "vol" de l'assertion "echantillon" en lui donnant la valeur calculée dans les antécédents.

- Nom de la règle : tm15.

- Antécédents : les antécédents "matchent" les substances contaminantes de l'échantillon, leur poids moléculaire et leur proportion.

- Conséquents : les conséquents modifient la caractéristique "contaminants" de l'assertion "echantillon", en lui donnant la valeur "nil". Ils créent dans la mémoire de travail une assertion dont le premier élément est l'atome identifiant "contaminant" et dont les éléments suivants sont le nom du contaminant, son poids moléculaire et sa proportion. Une telle assertion est créée pour chaque contaminant qui a satisfait les antécédents.

- Nom de la règle : tm16 et tm17.

- Antécédents : les antécédents "matchent" l'assertion "contaminant" créée par la règle précédente, la protéine à purifier et son poids moléculaire, le gel retenu pour la purification et son domaine de non-séparation. Ils vérifient que le poids moléculaire des contaminants est compris dans le domaine de non-séparation. Ils calculent pour les contaminants vérifiant ces conditions le pourcentage de contamination après application de la technique, en utilisant la primitive equation.

- Conséquents : les conséquents ajoutent à l'assertion "technique" sa caractéristique "gel" qui prend la valeur "type de gel" de l'assertion "gel" vérifiant les antécédents.

9.2.4.2. Evaluation de l'application de la technique

- Nom de la règle : tm13.

- Antécédents : les antécédents de cette règle "matchent" dans la mémoire de travail, la protéine à purifier et son domaine de stabilité, les tampons et leur(s) pk. Ils assurent que le domaine de stabilité de la protéine est compris dans un intervalle [pk-1, pk+1] pour chaque tampon, en utilisant la primitive compare. Ils ne gardent que les tampons dont l'intervalle [pk-1, pk+1] contient la valeur 7.4.

- Conséquent : le conséquent ajoute à l'assertion "technique" sa caractéristique "tampon" qui prend pour valeur la liste de noms de tampons qui ont satisfait les antécédents et sa caractéristique "pH" qui prend pour valeur 7.4..

- Nom de la règle : tm14.

- Antécédents : les antécédents matchent dans la mémoire de travail la caractéristique "volcol" de l'assertion "technique" et le volume de l'échantillon. Ils calculent à partir de là le volume de l'échantillon résultat.

- Conséquents : les conséquents retirent de la mémoire de travail, pour les contaminants qui vérifient les antécédents, l'assertion créée par la règle précédente. Ils créeent pour chacun de ces mêmes contaminants une assertion dont le premier élément est l'identifiant "element de contamination" et dont les éléments suivants sont le nom du contaminant et sa proportion.

- Nom de la règle : tm18.

- Antécédents : les antécédents "matchent" l'assertion "contaminant" créée par la règle tm15, la protéine à purifier et son poids moléculaire, le gel retenu pour la purification et son domaine de non-séparation. Ils vérifient que le poids moléculaire des contaminants n'est pas compris dans le domaine de non-séparation.

- Conséquents : les conséquents retirent de la mémoire de travail, pour les contaminants qui vérifient les antécédents, l'assertion créée par la règle tm15.

9.3. IMPLEMENTATION DE LA BASE DE META-REGLES

9.3.1. Méta-règles concernant la chromatographie sur échangeur d'ions

Nous présentons ici, en langage naturel, les règles de la base de méta-règles. Le lecteur intéressé trouvera le code de ces règles en annexe.

- Nom de la règle : mei1.

- Antécédents : les antécédents "matchent" l'assertion "ei envisageable" dans la mémoire de travail. Ils "matchent" les trois premières règles relatives à la technique dans la base de règles.

- Conséquents : les conséquents retirent de la mémoire de travail l'assertion "ei envisageable". Ils appliquent la primitive use-rule du moteur d'inférence à chaque règle qui a matché les antécédents.

Cette méta-règle implémente la tâche de choix d'application de la technique de la chromatographie sur échangeur d'ions.

- Nom de la règle : mei2.

- Antécédents : les antécédents "matchent" l'assertion "technique" dans la mémoire de travail et vérifient que sa caractéristique "nom" a reçu la valeur "ei". Ils "matchent" les règles ei4 à ei26 dans la base de règles.

- Conséquents : les conséquents appliquent la primitive use-rule du moteur d'inférence à chaque règle qui a matché les antécédents. Ils ajoutent à la mémoire de travail l'assertion "fin applicatoin ei".

Cette méta-règle implémente la tâche d'évaluation de l'application de la chromatographie sur échangeur d'ions.

- Nom de la règle : mei3.

- Antécédents : les antécédents "matchent" l'assertion "fin application ei" dans la mémoire de travail et ils la retirent. Ils "matchent" l'assertion "échantillon" et ses caractéristiques, ainsi que celles de l'assertion "technique". Ils présentent à l'utilisateur la technique envisagée et l'échantillon résultant de son application, en utilisant les primitives print-tech et print-echantillon. Ils assurent que l'utilisateur accepte ce qui lui est proposé.

- Conséquent : le conséquent crée l'assertion "ei accepte" dans la mémoire de travail.

Cette méta-règle implémente la présentation à l'utilisateur de la technique envisagée et l'évaluation du résultat de son application.

- Nom de la règle : mei4.

- Antécédents : les antécédents "matchent" l'assertion "ei accepte" ainsi que l'assertion "technique" et ses caractéristiques dans la mémoire de travail.

- Conséquents : les conséquents retirent l'assertion "ei accepte" et l'assertion "technique" de la base de faits. Ils créent les assertions "ei envisageable", "tm envisageable" et "pr envisageable".

- Nom de la règle : mei5.

- Antécédents : les antécédents "matchent" l'assertion "technique" et ses caractéristiques dans la mémoire de travail.

- Conséquent : le conséquent retire l'assertion "technique" de la base de faits. Il crée les assertions "ei envisageable", "tm envisageable" et "pr envisageable".

Cette règle n'est appliqueé que si la précédente n'a pas réussi.

Les méta-règles relatives à la chromatographie sur échangeur d'ions et à la précipitation isoélectrique sont très semblables aux méta-règles présentées ci-dessus. Elle ne seront donc pas détaillées ici. Pour la précipitation, le lecteur intéressé peut consulter l'annexe.

CONCLUSION

Evaluation du prototype [GAS83]

La technologie des systèmes experts en est encore à ses débuts. Il existe peu de techniques pour l'évaluation de tels systèmes, et celles qui existent sont assez primitives. Certes, il existe des critères tels que la correction du système, son efficacité, sa convivialité, uitilisés pour d'autres systèmes et appliqués aux systèmes experts. Mais les systèmes experts sont particuliers, dans le sens où ils contiennent de l'expertise humaine. Ils sont donc souvent comparés et évalués en regard des performances des humains. Dans un tel contexte, il n'est pas simple de savoir si une solution, qui est correcte pour le système expert, est celle que l'expert humain aurait fournie, celle qui satisferait un groupe d'experts ou celle qui serait une solution idéale pour tous. Personne ne peut vraiment évaluer la qualité d'une expertise humaine d'une manière adéquate et fiable. Il est encore plus ardu de le faire pour un système expert qui recrée cette expertise.

Lorsqu'on tente d'évaluer un système expert, les caractéristiques suivantes sont généralement prises en compte :

- la qualité des décisions et des conseils fournis par le système. Il n'est pas aisé d'évaluer ces caractéristiques, les systèmes experts étant précisément conçus pour des tâches où les décisions de l'expert humain sont plutôt subjectives et non-standardisées. Cependant, pour qu'un système expert soit accepté par ses destinataires potentiels, il est important que ceux-ci soient convaincus de la pertinence et de la fiabilité du conseil fourni par le système

- la correction des techniques de raisonnement utilisées. Il n'est pas absolument nécessaire que le mécanisme de raisonnement appliqué par le système soit le même que celui de l'expert humain, pourvu que le conseil fourni soit correct. Il peut cependant être utile d'y prêter quelqu'attention lors de l'évaluation du système, ce détail ayant parfois une influence sur la réalisation des autres caractéristiques
- la qualité des interactions homme-machine. Cela inclut des préocupations aussi diverses que le choix des mots utilisés dans le dialogue avec l'utilisateur, la capacité pour le système d'expliquer son raisonnement et de le façonner selon le degré d'expertise de l'utilisateur, la capacité à aider l'utilisateur lorsque ce dernier ne comprend pas ce que le système attend de lui, la capacité d'apprentissage pour l'utilisateur dans des termes qui lui sont familiers. De telles préocupations sont souvent déterminantes dans l'acceptation du système par l'utilisateur
- l'efficacité du système. L'impact d'un système expert sur le processus de prise de décision de l'utilisateur doit être également envisagé. Un système qui requiert un engagement en temps trop important de la part de l'utilisateur risque de ne pas être accepté, même s'il satisfait les autres exigences
- le coût de développement. S'il est prévu que le système expert soit un jour commercialisé, son évaluation au niveau du coût est aussi importante. Très peu de systèmes experts, actuellement, ont atteint ce stade.

L'évaluation d'un système par rapport à ces caractéristiques est un processus qui doit être continu. Il

doit commencer dès les premières étapes, que sont la définition des objectifs du système et la conception d'un prototype pour montrer la faisabilité, et se poursuivre jusqu'aux étapes terminales que sont la négociation des plans de maintenance et de mise à jour. Certaines de ces caractéristiques sont cependant moins adéquates à l'évaluation de certaines étapes que d'autres. Il est moins crucial de parler d'interface conviviale lors de la conception du premier prototype qu'à l'étape de raffinement du système.

La qualité des décisions prises par notre premier prototype n'a pu être testée. Nous avons soumis à notre système des exemples fictifs, créés par nous-mêmes. Les réponses fournies à partir de ces exemples sont conformes aux critères qui nous avaient été fournis par les experts. Il faudrait maintenant soumettre des cas réels, du moins des cas d'école, et montrer aux experts les réactions du système. Il est fort possible que les réponses du système, face à de tels cas, ne satisfassent pas les experts, les critères de raisonnement devant encore être affinés.

La correction des techniques de raisonnement n'a pas été envisagée. Cette caractéristique ne semble pas nécessaire pour un premier prototype, alors qu'il n'est même pas certain que les conseils fournis satisfassent les experts. Elle relève d'une interaction entre l'informatique et la psychologie, pour laquelle nous ne sommes certainement pas qualifiés.

Notre objectif était de montrer la faisabilité du développement d'un outil de type système expert dans le domaine. Il n'entre pas dans cet objectif de développer une interface conviviale. L'interface de notre système est assez dépouillée. Elle n'assure que le minimum pour rendre une interaction entre l'utilisateur et le système possible.

Du point de vue de l'efficacité, notre système demande encore un engagement assez important de la part de l'utilisateur. Lors de chaque utilisation, l'utilisateur doit, pour l'instant, préciser au système toutes les caractéristiques des substances contenues dans son échantillon. Cet inconvénient pourra cependant être corrigé assez rapidement. Il suffit que le système stocke toutes les données qui lui sont fournies sur les échantillons qu'il doit traiter. Le catalogue que le système propose à l'utilisateur avant de commencer une purification contiendra, de cette manière, de plus en plus souvent l'échantillon à purifier, ou, du moins, des substances de cet échantillon seront connues du système. Cette possibilité de stockage était jusqu'à présent superflue, puisque nous n'avons traité que des cas fictifs.

La question du coût de développement est dans notre cas tout à fait hors de propos, la commercialisation du système n'ayant jamais été envisagée.

Perspectives d'avenir

Il nous semble nécessaire, lors de développements futurs du système, de commencer par compléter la connaissance qui est à la base du processus de raisonnement. La base de connaissance actuelle représente une base raisonnable pour un premier prototype. Elle n'est cependant qu'une modélisation assez grossière du processus de raisonnement sous-jacent à la prise de décision par les experts.

Une première manière de compléter la base serait d'affiner les critères de raisonnement, notamment en ce qui concerne l'évaluation de la quantité de substances dans

l'échantillon après application d'une technique, la perte causée à l'échantillon par les techniques, le coût de chaque technique. Il faudrait également affiner la connaissance relative aux techniques annexes aux purifications.

On pourrait ensuite envisager de compléter la base par l'ajout de nouvelles techniques. Il faudrait, pour ces techniques, reprendre, avec les experts du domaine, tout le processus d'acquisition des connaissances.

La manière d'envisager les échantillons devrait sans doute être un peu élargie. Pour l'instant, les règles de choix d'une technique et d'évaluation de ses conséquences se basent uniquement sur des caractéristiques propres à l'échantillon, telles que son volume ou la quantité de protéines, ou sur des caractéristiques spécifiques aux substances qu'il contient, telles que le poids moléculaire ou le point isoélectrique. Il semble que ces caractéristiques soient souvent inconnues de l'expert, et à fortiori, des biochimistes en général. Par contre, ils connaissent souvent l'origine du mélange contenu dans l'échantillon. De telles connaissances devraient être mieux mises à profit. Le processus de raisonnement du système y gagnerait en finesse et en efficacité. Nous pensons, notamment, que le concept de mélange devrait être exploité davantage. D'autres caractéristiques doivent être dégagées qui permettraient de découvrir de nouvelles règles. Par exemple, un mélange contenant des résidus de cellules de membranes est trouble et ne peut être déposé sur une colonne. La purification n'est pas possible car la colonne se bouche. On peut alors penser à une précipitation isoélectrique. Il faut cependant savoir que l'agent de précipitation doit être l'acétone lorsque l'échantillon contient des cellules de membranes.

La mémoire de travail, nous l'avons déjà signalé dans le point précédent, doit elle aussi être enrichie.

Notre objectif était d'arriver à un premier prototype. Nous avons donc fait l'impasse sur les composants d'un système expert que sont l'interface, la capacité d'acquisition des connaissances, l'explication du raisonnement. Le développement de ce dernier composant, à partir du noyau que nous avons conçu et qui serait complété, transformerait le système en un outil d'enseignement. Cette voie d'avenir semble souhaitée tant par le directeur de notre mémoire que par le responsable du laboratoire. De plus, le manque de connaissances sur les caractéristiques des échantillons, déjà évoqué ci-dessus, fait que notre système serait beaucoup plus efficace en travaillant sur des cas d'école.

Problèmes rencontrés

La principale difficulté que nous avons rencontrée au cours de ce travail est la phase d'acquisition des connaissances. Bien que les experts du laboratoire aient fait preuve d'un grande disponibilité à notre égard, il leur était souvent assez difficile d'exprimer le raisonnement mis en oeuvre au cours d'une expérience de purification. Ils appliquent, parfois sans même s'en rendre compte, des critères acquis bien souvent par expérience. Cela leur a demandé un travail de réflexion sur leur propre domaine, en même temps qu'un effort d'explication pour rendre leur expertise accessible à nos faibles connaissances en biologie et en chimie. Pour notre part, nous avons dû nous familiariser avec leurs concepts et leur faire entrevoir les possibilités des systèmes experts, pour qu'ils comprennent mieux ce que nous attendions d'eux.

La seconde difficulté de taille est l'utilisation du langage LISP. Il s'agit d'un outil général qui permet de développer des mécanismes de raisonnement adaptés aux besoins des concepteurs. Il a cependant le désavantage que le concepteur doit partir de rien et développer lui-même un moteur d'inférence. L'utilisation d'un "shell" basé sur LISP aurait sans doute allégé notre travail de conception.

BIBLIOGRAPHIE

Domaine informatique

[BAR83] D.A. BARSTOW, N. AIELLO, R. DUDA, L.D. ERMAN, etc.,
Languages and Tools for Knowledge Engineering,
dans Building Expert Systems,
édité par F. HAYES-ROTH, D.A. WATERMAN, D.B. LENAT,
Addison-Wesley Pub. Comp., 1983, p.283 à 345.

[BON81] A. BONNET,
Applications de d'Intelligence Artificielle : les Systèmes experts,
RAIRO Informatique, vol.15, n°4, 1981, p. 325 à 362.

[BRA83] R.J. BRACHMAM, S. AMAREL, C. ENGELMAN, R. ENGELMORE, E.A. FEIGENBAUM, D.E. WILKINS,
What Are Experts Systems ?,
dans Building Expert Systems,
édité par F. HAYES-ROTH, D.A. WATERMAN, D.B. LENAT,
Addison-Wesley Publishing Company, 1983, p.31 à 57.

[BRO85] L. BROWNSTON, R. FARRELL, E. KANT, N. MARTIN,
Programming Expert Systems in OPS5 - An introduction to rule-based programming,
Addison-Wesley Pub. Comp., 1985.

[BUC83] B.G. BUCHANAN, D. BARSTOW, R. BECHTAL, etc.,
Constructing an Expert System,
dans Building Expert Systems,
édité par F. HAYES-ROTH, D.A. WATERMAN, D.B. LENAT,
Addison-Wesley Pub. Comp., 1983, p.127 à 167.

[BUC84] B.G. BUCHANAN et E.H. SHORTILIFFE,
Rule-Based Expert System - the Mycin Experiments of the Stanford Heuristic Programming Project,
Addison-Wesley Publishing Company, 1984.

[CHA84] E. CHARNIAK et D. McDERMOTT,
Introduction to Artificial Intelligence,
Addison-Wesley Publishing Company, 1984.

[CLA87] W.J. CLANCEY,
Knowledge-Based Tutoring, The Guidon Program,
MIT Press, 1987.

[DAV81] R. DAVIS et D.B. LENAT,
Knowledge-Based Systems in Artificial Intelligence,
Mac Graw-Hill, Inc., 1982.

[FEI79] E.A. FEIGENBAUM,
Themes and Case Studies of Knowledge Engineering,
dans Expert Systems in the Micro-Electronic Age,
Edinburgh University Press, 1979.

[GAN85] J-G. GANASCIA,
La conception des systèmes experts,
dans La Recherche en Intelligence Artificielle,
Editions du Seuil, La Recherche, 1987, p.313 à 334.

[GAS83] J. GASCHNIG, P. KLAHR, H. POPLE, E. SHORTLIFFE, A. TERRY,
Evaluation of Expert Systems : Issues end Cases Studies,
dans Building Expert Systems,
Addison-Wesley Publishing Company, 1983,
p.241 à 280.

[HAY83] F. HAYES-ROTH, D.A. WATERMAN, D.B. LENAT,
An Overview of Expert Systems,
dans Building Expert Systems,
Addison-Wesley Publishing Company, 1983, p.3 à 29.

[HAY84] F. HAYES-ROTH,
The Knowledge-Based Expert System : A Tutorial,
Computer, IEEE, Septembre 1984, p. 11 à 28.

[JAC87] P. JACKSON,
Towards an architecture for advice-giving systems,
dans Current Issues in Expert Systems,
edité par A. VAN LAMSWEERDE et P. DUFOUR ,
Academic Press Inc., 1987.

[KLA86] P.KLAHR et D.A. WATERMAN,
Expert Systems, Techniques, Tools and Applications,
Addison-Wesley Publishing Company, 1986.

[LAU82] J-L. LAURIERE,
Représentation et utilisation des connaissances; deuxième partie : Représentation des connaissances,
Techniques et Sciences Informatiques, vol.1, n°2,
1982, p. 109 à 133.

[NIL71] N.J. NILSSON,
Problem-Solving Methods in Artificial Intelligence,
McGraw-Hill, 1971.

[PIN81] S. PINSON,
Représentation des Connaissances dans les Systèmes Experts,
RAIRO Informatique, vol.15, n°4, 1981, p. 343 à 362.

[RIC83] E. RICH,
Artificial Intelligence,
McGraw-Hill, 1983.

[STE82] M. STEFIK, J. AIKINS, R. BALSER, J. BENOIT,
L. BIRNBAUM, F. HAYES-ROTH, E. SACERDOTI,
The Organization of Expert Systems, A Tutorial,
Artificial Intelligence, North-Holland Publishing,
1982, p. 135 à 173.

[WAT86] D.A. WATERMAN,
A Guide to Expert Systems,
Addison-Wesley Publishing Company, 1986.

[WIN84] P.H. WINSTON et B.K. HORN,
Lisp,
Addison-Wesley Publishing Company, 1984.

[YUA87] T. YUASA et M. HAGIYA,
Introduction to Common Lisp,
Academic Press Inc., 1987.

muLISP Reference Manual, MICROSOFT LISP, Artificial Intelligence Programming Environment for the MS-DOS Operating System,
Microsoft Corporation, 1986.

Domaine biochimique

[COH43] E. COHN, J. T. EDSALL,
Proteins, amino acids and peptides,
Academic Press, New York, 1943.

[FIS80] L. FISCHER,
Gel filtration chromatography,
Elsevier, North-Holland Biomedical Press,1980.

[PHAa] Sephadex-Gel Filtration in Theory and Pratice,
Pharmacia Fine Chemicals, Uppsala, Sweden, ?.

[PHAb] Ion Exchange Chromatography principles and methods,
Pharmacia Fine Chemicals, Uppsala, Sweden, ?.

[REM86] J. REMACLE,
Techniques biochimiques,
Facultés Universitaires Notre-Dame de la Paix,
Namur, 1986.

[ROG85] M. ROGER,
Mise au point d'un biosenseur pour le dosage du NADH,
Facultés Universitaires Notre-Dame de la Paix, Namur, 1985.

ANNEXE

CODE DU PROGRAMME

```
(loop (prin1 '*) (eval (read)) ((null rds)))

(defun proteins ()
   (setq facts (load-memory "facts")) (rds)
   (let ((asserts (sample-to-purify)))
        (setq assertions (car asserts))
        (setq newfacts (cadr asserts))
   )
   (setq meta-rules (load-memory "meta")) (rds)
   (setq rules (load-memory "rules")) (rds)
   (setq assertions (append '((ei envisageable)
                              (tm enviseagle)
                              (pr enviseable)
                             )
                             assertions
                    )
   )
   (forward-chaining meta-rules)
   (setq facts
         (acquisition-memory
                      newfacts
                      facts
                      '((colonne (diametre hauteur))
                                 (gel type)
                                 (tampon type)
                                 (melange nom)
                                 (substance nom)
                       ))
   )
   (save-memory "facts" facts)
)

; MOTEUR D'INFERENCE

;  manipulation de la liste d'assertions [9.1.1.]

(defun add-assertion (assert)
   ((member assert assertions 'equal) assert)
   ((setq assertions (cons assert assertions))
    assert
   )
)
(defun remove-assertion (assert)
   ((setq assertions (remove assert assertions 'equal))
     assert
   )
)

; "pattern-matching" symbolique [9.1.2.]
```

```
(defun pattern-variable (lst) (cadr lst))
(defun shove-pair (var val lstasso)
   ((null lstasso) (list (list var val)))
   ((equal var (caar lstasso))
    (cons (list var val) (cdr lstasso))
   )
   (cons (car lstasso) (shove-pair var
                                   val
                                   (cdr lstasso)
   )                   )
)
(defun shove-group (var val lstasso)
   ((null lstasso) (list (list var (list val))))
   ((equal var (caar lstasso))
    (cons (list var (append (cadar lstasso) (list val)))
          (cdr lstasso)
    )
   )
   (cons (car lstasso) (shove-group var
                                    val
                                    (cdr lstasso)
   )                   )
)
(defun shove-groups (var lstval lstasso)
   ((null lstval) lstasso)
   (shove-groups var (cdr lstval)
                 (shove-group var (car lstval) lstasso)
   )
)
(defun pull-value (var lstasso)
   (cadr (assoc var lstasso))
)
(defun match (pattern assertion lstasso)
   ((and (null pattern) (null assertion))
    (((null lstasso))
     lstasso
    )
   )
   ((null pattern) nil)
   ((or (equal (car pattern) '?)
        (equal (car pattern) (car assertion))
    )
    (match (cdr pattern) (cdr assertion) lstasso)
   )
   ((equal (car pattern) '+)
    (if (equal assertion nil)
        (if (equal (cdr pattern) nil)
            (if (null lstasso) t lstasso)
            nil
        )
        (or (match (cdr pattern) (cdr assertion) lstasso)
            (match (cdr pattern) assertion lstasso)
            (match pattern (cdr assertion) lstasso)
```

```
      )
 )
)
((equal (pattern-indicator (car pattern)) '>>)
 (match (cdr pattern) (cdr assertion)
        (shove-pair (pattern-variable (car pattern))
                    (car assertion)
                    lstasso
        )
 )
)
((equal (pattern-indicator (car pattern)) '>+)
 (let ((new-lstasso
        (shove-group (pattern-variable (car pattern))
                     (car assertion)
                     lstasso
        )
       ))
      (or (match (cdr pattern) (cdr assertion)
                 new-lstasso
          )
          (match pattern (cdr assertion) new-lstasso)
      )
 )
)
((equal (pattern-indicator (car pattern)) '<<)
 (match (cons (pull-value
                    (pattern-variable (car pattern))
                    lstasso
              )
              (cdr pattern)
        )
        assertion
        lstasso
 )
)
((equal (pattern-indicator (car pattern)) '<+)
 (match (append (pull-value
                     (pattern-variable (car pattern))
                     lstasso
                )
                (cdr pattern)
        )
        assertion
        lstasso
 )
)
((and (listp (car pattern))
      (consp (match (car pattern)
                    (car assertion)
                    lstasso
             )
      )
```

```
     )
     (match (cdr pattern) (cdr assertion)
            (match (car pattern) (car assertion) lstasso)
     )
    )
)

; manipulation des operations [9.1.3]

(defun operation-indicator (lst) (car lst))
(defun operation-function (lst) (cadr lst))
(defun replace-variables (lstargu lstasso)
   ((atom lstargu) lstargu)
   ((or (equal (pattern-indicator lstargu) '<<)
        (equal (pattern-indicator lstargu) '<+)
    )
    (cadr (assoc (pattern-variable lstargu) lstasso))
   )
   ((equal (operation-indicator (car lstargu)) 'ope)
    (apply (operation-function (car lstargu))
           (replace-variables (cddr lstargu) lstasso)
    )
   )
   (if (equal (pattern-indicator (car lstargu)) '<+)
       (append (replace-variables (car lstargu) lstasso)
               (replace-variables (cdr lstargu) lstasso)
       )
       (cons (replace-variables (car lstargu) lstasso)
             (replace-variables (cdr lstargu) lstasso)
       )
   )
)
(defun apply-operation (operation lstasso)
   ((equal (pattern-indicator (cadadr operation)) '>>)
    (let ((result (apply (operation-function
                                  (car operation)
                         )
                         (replace-variables
                                 (cddr operation)
                                 lstasso
                         )
                  )
          ))
          ((and (equal (caddr (cadr operation)) '(nonil))
                (neq result nil)
           )
           (shove-pair (pattern-variable
                               (cadadr operation)
                       )
                       result
                       lstasso
           )
          )
```

```
          ((and (not (equal (caddr (cadr operation)) nil))
                (equal result (caaddr (cadr operation)))
           )
           (shove-pair (pattern-variable
                                (cadadr operation)
                       )
                       result
                       lstasso
           )
          )
          ((equal (caddr (cadr operation)) nil)
           (shove-pair (pattern-variable
                                (cadadr operation)
                       )
                       result
                       lstasso
           )
          )
          nil
     )
    )
    ((equal (cadadr operation) nonil)
     (not (equal (apply (operation-function
                                  (car operation)
                        )
                        (replace-variables
                                (cddr operation) lstasso
                        )
                )
                nil
          )
     )
    )
    ((equal (cadadr operation)
            (apply (operation-function (car operation))
                   (replace-variables (cddr operation)
                                      lstasso
                   )
            )
     )
    )
)

; manipulation des acces [9.1.4.]

(defun access-indicator (lst) (car lst))
(defun access-identificator (lst) (cadr lst))
(defun apply-access (access lstasso)
   (let ((stream (filter-lstasso
                  (list (access-identificator
                                (car access)
                        )
                        (list '>+ 'v)
```

```
                    )
                    assertions
                  )
          ))
          (remove-assertion
             (build-assertion
                   (access-identificator (car access))
                   (cadaar stream)
          )  )
          (add-assertion
              (build-assertion
                  (access-identificator (car access))
                  (cadaar stream)
                  (replace-variables (cdr access)
                                     lstasso
                  )
             )
          )
    )
)
(defun build-assertion (id oldatt newatt)
    ((null newatt) (cons id oldatt))
    ((eq (caar newatt) 'm)
     (build-assertion id
                      (shove-pair (cadar newatt)
                                  (car (caddar newatt))
                                  oldatt
                      )
                      (cdr newatt)
     )
    )
    ((eq (caar newatt) 'a)
     (build-assertion id
                      (shove-groups (cadar newatt)
                                    (caddar newatt)
                                    oldatt
                      )
                      (cdr newatt)
     )
    )
)

; manipulation des streams [9.1.6.]

(defun combine-streams (s1 s2) (append s1 s2))
(defun add-to-stream (e s) (cons e s))
(defun first-of-stream (s) (car s))
(defun rest-of-stream (s) (cdr s))
(defun empty-stream-p (s) (null s))
(defun make-empty-stream () nil)

; moteur d'inference [9.1.7.4.]
```

```
(defun filter-lstasso (filter base lstasso)
   ((equal (operation-indicator (car filter)) 'ope)
    (let ((result (apply-operation filter lstasso)))
         ((eq result t) (list lstasso))
         ((eq result nil) nil)
         (list result)
    )
   )
   ((null base) (make-empty-stream))
   (let ((new-list (match filter (car base) lstasso)))
        ((not (equal new-list nil))
         (add-to-stream new-list
                        (filter-lstasso filter (cdr base)
                                        lstasso
                        )
         )
        )
        (filter-lstasso filter (cdr base) lstasso)
   )
)
(defun filter-stream (filter stream)
   ((empty-stream-p stream) (make-empty-stream))
   ((equal (caar filter) 'rule)
    (combine-streams (filter-lstasso
                           (cadr filter)
                           rules
                           (first-of-stream stream)
                     )
                     (filter-stream
                           filter
                           (rest-of-stream stream)
                     )
    )
   )
   (combine-streams (filter-lstasso
                          filter
                          assertions
                          (first-of-stream stream)
                    )
                    (filter-stream
                          filter
                          (rest-of-stream stream)
                    )
   )
)
(defun cascade-through-filters (filters stream)
   ((null filters) stream)
   ((filter-stream
           (car filters)
           (cascade-through-filters (cdr filters) stream)
   ))
)
(defun execute-action (action lstasso)
```

```
   ((equal (operation-indicator (car action)) 'ope)
   (if (apply-operation action lstasso) (list 'ope) nil))
   ((equal (access-indicator (car action)) 'id)
    (if (apply-access action lstasso) (list 'id) nil)
   )
   (add-assertion (replace-variables action lstasso))
)
(defun spread-through-actions (actions lstasso)
   ((null actions) (make-empty-stream))
   (add-to-stream (execute-action (car actions) lstasso)
                  (spread-through-actions (cdr actions)
                                          lstasso
                  )
   )
)
(defun feed-to-actions (actions stream)
   ((empty-stream-p stream) (make-empty-stream))
   (combine-streams (spread-through-actions
                           actions
                           (first-of-stream stream)
                    )
                    (feed-to-actions
                           actions
                           (rest-of-stream stream)
                    )
   )
)
(defun use-rule (rule)
   (let* ((rule-name (cadr rule))
          (ifs (reverse (cdr (caddr rule))))
          (thens (cdr (cadddr rule)))
          (stream (cascade-through-filters
                    ifs
                    (add-to-stream
                           nil
                           (make-empty-stream)
                    )
                  )
          )
          (action-stream (feed-to-actions thens stream))
         )
         (not (empty-stream-p action-stream))
   )
)
(defun forward-chaining (rules-to-try)
   ((null rules-to-try))
   ((use-rule (car rules-to-try))
    (forward-chaining meta-rules)
   )
   (forward-chaining (cdr rules-to-try))
)
```

```
; PRIMITIVES UTILISEES PAR LES REGLES [9.2.1.]

; fonctions de calcul

(defun diameter (vol heig)
   (truncate
    (* (sqrt (truncate (/ vol (* heig (pi) 2)))) 2)
   )
)
(defun equation (x0 y0 x1 y1 x)
   (- y1 (* (/ (- x x0) (- x1 x0)) (- y1 y0)))
)

; fonctions de manipulation de listes

(defun sub-1 (lst) (mapcar 'sub1 lst))
(defun add-1 (lst) (mapcar 'add1 lst))
(defun compare (ope elem lst)
   ((null lst) nil)
   ((equal (apply ope (list elem (car lst))) t)
    (cons (car lst) (compare ope elem (cdr lst)))
   )
   (compare ope elem (cdr lst))
)
(defun succession (l1 l2 l3 l4)
   ((null l1) l4)
   ((or (equal l2 nil) (equal l3 nil)) l4)
   ((and (= (car l1) (+ (car l2) 1))
         (= (car l1) (- (car l3) 1))
    )
    (cons (car l1)
          (succession (cdr l1) (cdr l2) (cdr l3) l4)
    )
   )
   ((= (car l1) (+ (car l2) 1))
    (succession (cdr l1) (cdr l2) l3 l4)
   )
   ((= (car l1) (- (car l3) 1))
    (succession (cdr l1) l2 (cdr l3) l4)
   )
)
(defun nearest (dir pi tampons)
   ((eq dir below)
    (car (near-below pi tampons '(nil 20)))
   )
   ((eq dir above)
    (car (near-above pi tampons '(nil 20)))
   )
)
(defun near-below (pi tamps tampons)
   ((null tamps) tampons)
   ((< (- (caadar tamps) pi 1) (cadr tampons))
    (near-below pi (cdr tamps)
```

```
                    (list (list (caar tamps))
                          (- (caadar tamps) pi 1)
                    )
     )
    )
    ((= (- (caadar tamps) pi 1) (cadr tampons))
     (near-below pi (cdr tamps)
                    (list (cons (caar tamps)
                                (car tampons)
                          )
                          (cadr tampons)
                    )
     )
    )
    (near-below pi (cdr tamps) tampons)
)
(defun near-above (pi tamps tampons)
    ((null tamps) tampons)
    ((< (- pi (+ (car (last (cadar tamps))) 1))
        (cadr tampons)
     )
     (near-above pi (cdr tamps)
                 (list (list (caar tamps))
                       (- pi
                          (+ (car (last (cadar tamps)))
                             1
                          )
                       )
                 )
     )
    )
    ((= (- pi (+ (car (last (cadar tamps))) 1))
        (cadr tampons)
     )
     (near-above pi (cdr tamps) (list (cons (caar tamps)
                                            (car tampons)
                                      )
                                      (cadr tampons)
                                )
     )
    )
    (near-above pi (cdr tamps) tampons)
)

; fonctions d'interface

(defun print-tech (technique)
    (clear-screen)
    (write-string "Technique envisagee : ")
    (print-title-of-tech technique)
    (print-elems-of-concept
      '(d h ph gel volgel tampon t elution perte
        volagprec)
```

```
      '(si si si sc si rc si sc si si)
      '("diametre de la colonne" "hauteur de la colonne"
        "pH de purification : " "gel pour la purification"
        "volume de gel" "tampon" "duree d'application"
        "type d'elution"  "perte d'activite"
        "volume de l'agent de precipitation"
       )
       technique
       22
   )
)
(defun print-sample (sample)
   (terpri 2)
   (write-string "echantillon obtenu : ")
   (print-elems-of-concept
      '(melange vol q cc proteine contaminants)
      '(sc si si si sc rdi)
      '("nom du melange" "volume : " "quantite"
        "concentration" "proteine a purifier"
        "contaminants"
        (nil "dans une proportion de")
       )
       sample
       22
   )
   (terpri)
   (o-ou-n
     "Acceptez-vous l'application de la technique ? "
   )
)
(defun print-title-of-tech (technique)
   ((eq (cadr (assoc 'nom technique)) 'ei)
    (write-line "chromatographie sur echange d'ions")
   )
   ((eq (cadr (assoc 'nom technique)) 'tm)
    (write-line "chromatographie sur tamis moleculaire")
   )
   ((eq (cadr (assoc 'nom technique)) 'p)
    (write-line "precipitation isoelectrique")
   )
)
(defun question-height (height)
   (terpri)
   (loop
     (setq height
           (string-number (question-response
             "Hauteur de la colonne ? (entre 10 et 20cm)"
           )                )
     )
     ((and (>= height 10) (<= height 20)) height)
   )
)
(defun question-decroch (char read-char)
```

```
    (terpri)
    (write-string
      "Decrochage par gradient ou par etapes ? (g ou e) "
    )
    (clear-input)
    (loop
      (setq char (string-upcase (read-char)))
      ((or (eq char 'g) (eq char 'e))
       (write-line (string-downcase char))
       char
      )
      (write-byte 7)
    )
)

; DETERMINATION DE L'ECHANTILLON ET CONSTRUCTION DE LA
; BASE DE FAITS

; chargement et sauvetage des connaissances

(defun load-memory (fil base)
   (rds (pack* "b:" fil ".mem"))
   (read)
)
(defun save-memory (fil base)
   (wrs (pack* "b:" fil ".mem"))
   (print-base base 0)
   (wrs)
)
(defun print-base (base tab)
   (princ '"(") (terpri)
   (print-elems-of-base base (add1 tab))
   (princ '")") (terpri)
)
(defun print-elems-of-base (base tab)
   ((null base))
   (spaces tab)
   (princ (pack* '"(" (caar base))) (terpri)
   (print-assertions-of-base (cdar base) (+ tab 2))
   (spaces tab)
   (princ '")") (terpri)
   (print-elems-of-base (cdr base) tab)
)
(defun print-assertions-of-base (asserts tab)
   ((null asserts))
   (spaces tab)
   (princ (car asserts)) (terpri)
   (print-assertions-of-base (cdr asserts) tab)
)

; determination de l'echantillon et acquisition des
; connaissances aupres de l'utilisateur
```

```
(defun sample-to-purify ()
   (let* ((components (mixture-in-sample))
          (properties (components-in-mixture
                       (cdr components))
          )
          (volume (acquisition-volume))
          (quantity (acquisition-quantity))
          (concentration (acquisition-concentration))
          (protein (acquisition-protein))
          (contaminates
             (acquisition-contaminates protein
                                       (cdr components)
             )
          )
          (assertions (build-assertions components
                                        properties
                                        volume
                                        quantity
                                        concentration
                                        protein
                                        contaminates
          )           )
          (newassertions
            (append
              (build-concept 'substance
                             properties
              )
              (list (list 'melange
                          (list 'nom (car components))
                          (list 'substances
                                (cdr components)
                          )
          ) ) )     )
         )
         (list assertions newassertions)
   )
)
(defun mixture-in-sample ()
   (let* ((lstmixture
            (cadaar (filter-lstasso
                       '(melange (>+ mixture))
                         facts
          ) )       )
          (catalogue (build-lst-properties 'nom
                                           lstmixture
          )          )
          (mixture (print-catalogue catalogue))
          (components
            (property-of-concept 'substances
                                 lstmixture
                                  mixture
          ) )
         )
```

```
         (cons mixture (extent-mixture components
                                       mixture
         )             )
   )
)
(defun extent-mixture (lstcomponents mixture)
   (print-mixture lstcomponents mixture)
   ((not (question-acquisition-concept)) lstcomponents)
   (acquisition-char
       (print-mixture (acquisition-mixture lstcomponents)
                      mixture
       )
   )
)
(defun components-in-mixture (components)
   (let ((keys '(nature pm pi domsta dompre duree-sta))
         (msgs '("nature" "masse moleculaire"
                 "point isoelectrique"
                 "domaine de stabilite"
                 ("borne inferieure" "borne superieure")
                 "domaine de precipitation"
                 ("borne inferieure" "borne superieure")
                 "duree de stabilite"
                )
         )
         (types '(sc si si di di si))
         (lstcomponents
            (cadaar (filter-lstasso
                           '(substance (>+ substances))
                           facts
            )       )
         )
        )
        (extent-components lstcomponents components keys
                           msgs types
        )
   )
)
(defun build-assertions
            (components properties vol q cc prot cont)
   (append
      (build-concept
        'colonne
        (cadaar (filter-lstasso '(colonne (>+ colonne))
                                facts
        )       )
      )
      (build-concept
         'gel
         (cadaar (filter-lstasso '(gel (>+ gel)) facts))
      )
      (build-concept
         'tampon
```

```
          (cadaar (filter-lstasso '(tampon (>+ tampon))
                                  facts
                  )
          )
       )
       (build-concept 'substance properties)
       (list (list 'echantillon
                   (list 'melange (car components))
                   (list 'vol vol)
                   (list 'q q)
                   (list 'cc cc)
                   (list 'proteine prot)
                   (list 'contaminants cont)
       )     )
   )
)
(defun build-concept (concept l1 l2)
   ((null l1) l2)
   (cons (cons concept (car l1))
         (build-concept concept (cdr l1) l2)
   )
)
(defun build-string (msgs lst)
   ((null lst) '"")
   ((null (car msgs))
    (pack* (car lst) " "
           (build-string (cdr msgs) (cdr lst))
    )
   )
   (pack* (car msgs) " " (car lst)
          " " (build-string (cdr msgs) (cdr lst))
   )
)
(defun build-lst-properties (property lstmixture)
   ((null lstmixture) nil)
   (cons (give-property property (car lstmixture))
         (build-lst-properties property (cdr lstmixture))
   )
)
(defun give-property (key lstasso)
   (cadr (assoc key lstasso))
)
(defun property-of-concept (property lstconcept concept)
   ((null lstconcept) nil)
   ((equal (give-property 'nom (car lstconcept)) concept)
    (give-property property (car lstconcept))
   )
   (property-of-concept property
                        (cdr lstconcept) concept
   )
)
(defun search-lstasso (lstassos elem)
   ((null lstassos) nil)
```

```
   ((equal (give-property 'nom (car lstassos)) elem)
           (car lstassos)
   )
   (search-lstasso (cdr lstassos) elem)
)
(defun replace-lstasso (lstasso lstassos elem)
   ((null lstassos)
    (list (cons (list 'nom elem) lstasso))
   )
   ((equal (give-property 'nom (car lstassos)) elem)
    (cons  lstasso (cdr lstassos))
   )
   (cons (car lstassos)
         (replace-lstasso lstasso (cdr lstassos) elem)
   )
)
(defun join (l1 l2)
   ((null l2) l1)
   (join (shove-pair (caar l2) (cadar l2) l1) (cdr l2))
)

; sauvetage des connaissances acquises aupres de
; l'utilisateur

(defun acquisition-memory (lstnewfacts lstoldfacts ident)
   ((null lstnewfacts) lstoldfacts)
   (let ((oldfacts
            (cadaar (filter-lstasso
                           (list (caar lstnewfacts)
                           '(>+ var))
                           lstoldfacts
            )       )
         )
         (newfact (cdar lstnewfacts))
         (identificator
              (give-property (caar lstnewfacts) ident)
         )
        )
        (acquisition-memory
           (cdr lstnewfacts)
           (build-facts (cons (caar lstnewfacts)
                              (build-fact newfact
                                          oldfacts
                                          identificator
                              )
                        )
                        lstoldfacts
           )
           ident
        )
   )
)
(defun build-facts (lstfacts facts)
```

```
   ((null facts) (list lstfacts))
   ((equal (car lstfacts) (caar facts))
    (cons lstfacts (cdr facts))
   )
   (cons (car facts) (build-facts lstfacts (cdr facts)))
)
(defun build-fact (fact lstoldfacts identificator)
   ((null lstoldfacts) (list fact))
   ((identical fact (car lstoldfacts) identificator)
    (cons fact (cdr lstoldfacts))
   )
   (cons (car lstoldfacts)
         (build-fact fact (cdr lstoldfacts)
                     identificator
         )
   )
)
(defun identical (fact oldfact identificator)
   ((null identificator) t)
   ((not (listp identificator))
    (equal (give-property identificator fact)
           (give-property identificator oldfact)
    )
   )
   ((identical fact oldfact (car identificator))
    (identical fact oldfact (cdr identificator))
   )
   nil
)
```

```
; fonctions d'interface

(defun print-catalogue (catalogue)
   (clear-screen)
   (set-cursor 0 25)
   (write-line "CATALOGUE DES MELANGES")
   (set-cursor 1 25)
   (write-line "----------------------")
   (print-mixtures catalogue)
   (terpri 2)
   (question-response "Quel est le nom du melange ? ")
)
(defun print-mixtures (mixtures row column)
   (if (not (numberp row)) (setq row 3))
   (if (not (numberp column)) (setq column 5))
   ((null mixtures) (terpri 2))
   (set-cursor row column)
   (print (car mixtures))
   (if (eq column 55) (setq column -20))
   (if (eq column -20) (setq row (+ row 1)))
   (print-mixtures (cdr mixtures) row (+ column 25))
)
(defun print-mixture (components mixture)
   ((null components)
    (write-line
      (pack* "Le melange " mixture " est inconnu !")
    )
    nil
   )
   (clear-screen)
   (write-line (pack* "Le melange " mixture " contient"))
   (terpri)
   (print-components-of-mixture components)
   components
)
(defun acquisition-mixture (components)
   (terpri)
   (write-line "Entrez les substances inconnues")
   (write-line
      "Corrigez la proportion des substances connues"
   )
   (join components (acquisition-components-of-mixture))
)
(defun acquisition-components-of-mixture ()
   (terpri)
   (let ((substance (question-response
                      "nom de la substance : ")
        ))
        (if (not (equal substance '""))
            (cons (list substance
                        (string-number (question-response
                          "proportion dans le melange : "
                        ))
```

```
                    )
                    (acquisition-components-of-mixture)
                )
                nil
            )
        )
    )
    (defun extent-components
                (lstcomponents components keys msgs types)
        ((null components) lstcomponents)
        (clear-screen)
        (write-line (pack* "caracteristiques de "
                           (caar components)
                    )
        )
        (terpri)
        (print-elems-of-concept keys
                                types
                                msgs
                                (search-lstasso
                                       lstcomponents
                                       (caar components)
                                )
                                22
                                t
        )
        ((not (question-acquisition-concept))
         (extent-components lstcomponents (cdr components)
                            keys msgs types
         )
        )
        (let ((newcomponent (acquisition-properties-of-element
                                    keys
                                    msgs
                                    (search-lstasso
                                           lstcomponents
                                           (caar components)
                                    )
                                    types
                                    (caar components)
                            )
             ))
             (clear-screen)
             (write-line (pack* "caracteristiques de "
                                (caar components)
             )           )
             (terpri)
             (print-elems-of-concept keys types msgs
                                     newcomponent 22 t)
             (acquisition-char nil)
             (extent-components (replace-lstasso
                                        newcomponent
                                        lstcomponents
```

```
                                        (caar components)
                                    )
                                    (cdr components)
                                    keys
                                    msgs
                                    types
            )
        )
    )
    (defun acquisition-volume ()
        (clear-screen)
        (string-number
          (question-response
            "Quel est le volume de l'echantillon ? "))
    )
    (defun acquisition-quantity ()
        (terpri)
        (string-number
          (question-response
           "Quelle est la quantite totale de l'echantillon ? "
        ))
    )
    (defun acquisition-concentration ()
        (terpri)
        (string-number
          (question-response
            "Quelle est la concentration de l'echantillon ? "
        ))
    )
    (defun acquisition-protein ()
        (terpri)
        (question-response
           "Quelle est la substance a purifier ? ")
    )
    (defun acquisition-contaminates (protein components)
        (terpri)
        (write-line "Quels sont les contaminants ? ")
        (acquisition-element-of-contamination protein
                                              components
        )
    )
    (defun acquisition-element-of-contamination
           (protein components)
        ((null components) nil)
        ((equal (caar components) protein)
         (acquisition-element-of-contamination protein
                                              (cdr components)
         )
        )
        ((o-ou-n (pack* "La substance " (caar components)
                        " est-elle un contaminant ? ")
         )
         (cons (car components)
```

```
            (acquisition-element-of-contamination protein
                        (cdr components))
      )
     )
     (acquisition-element-of-contamination protein
                 (cdr components)
     )
)
(defun acquisition-properties-of-element
        (keys msgs lstasso types)
    ((null keys) lstasso)
    (terpri)
    ((or (equal (car types) 'sc) (equal (car types) 'si))
     ((o-ou-n
       (pack* "Voulez-vous modifier " (car msgs) " ? ")
      )
      (let ((response
             (if (equal (car types) 'si)
                 (string-number (question-response
                       (pack* (car msgs) " : ")
                 )               )
                 (question-response
                          (pack* (car msgs) " : ")
                 )
             )
            ))
            (acquisition-properties-of-element
                      (cdr keys)
                      (cdr msgs)
                      (shove-pair (car keys)
                                  response lstasso
                      )
                      (cdr types)
            )
      )
     )
     (acquisition-properties-of-element (cdr keys)
                 (cdr msgs) lstasso (cdr types)
     )
    )
    ((or (equal (car types) 'dc) (equal (car types) 'di))
     ((o-ou-n (pack*
        "Voulez-vous modifier " (car msgs) " ? ")
      )
      (acquisition-properties-of-element
                 (cdr keys)
                 (cddr msgs)
                 (shove-pair (car keys)
                             (acquisition-elems-of-lst
                                     (cadr msgs)
                                     (car types)
                             )
                             lstasso
```

```
                    )
                    (cdr types)
          )
         )
         (acquisition-properties-of-element (cdr keys)
                    (cddr msgs) lstasso (cdr types)
         )
        )
        ((or (equal (car types) 'rc) (equal (car types) 'ri))
         ((o-ou-n (pack*
            "Voulez-vous modifier " (car msgs) " ? ")
          )
          (acquisition-properties-of-element
                    (cdr keys)
                    (cdr msgs)
                    (shove-pair (car keys)
                                (acquisition-elems-of-elem
                                   (car msgs)
                                   (car types)
                                )
                                lstasso
                    )
                    (cdr types)
          )
         )
         (acquisition-properties-of-element (cdr keys)
                       (cdr msgs) lstasso (cdr types)
         )
        )
    )
    (defun print-elems-of-concept
           (elems types msgs lstasso col inc)
       ((null elems))
       ((or (equal (car types) 'sc) (equal (car types) 'si))
        (let ((elem (cadr (assoc (car elems) lstasso))))
             (set-cursor (row) col)
             ((null elem)
              (if inc
                  (write-line (pack* (car msgs)
                     " : caracteristique inconnue"))
              )
              (print-elems-of-concept (cdr elems) (cdr types)
                                      (cdr msgs) lstasso col inc
              )
             )
             (write-line (pack* (car msgs) " : "
                        (string-downcase elem))
             )
             (print-elems-of-concept (cdr elems) (cdr types)
                                     (cdr msgs) lstasso col inc
             )
        )
       )
```

```
((or (equal (car types) 'dc) (equal (car types) 'di))
 (let ((elem (cadr (assoc (car elems) lstasso))))
      (set-cursor (row) col)
      ((null elem)
       (if inc
           (write-line (pack* (car msgs)
                 " : caracteristique inconnue")
           )
       )
       (print-elems-of-concept (cdr elems) (cdr types)
                          (cddr msgs) lstasso col inc
       )
      )
      (write-line (car msgs))
      (print-elems-of-lst (cadr msgs)
                          elem
                          (row)
                          (+ col 5)
      )
      (print-elems-of-concept (cdr elems)
                              (cdr types)
                              (cddr msgs)
                              lstasso
                              col
                              inc
      )
 )
)
((or (equal (car types) 'rc) (equal (car types) 'ri))
 (let ((elem (cadr (assoc (car elems) lstasso))))
      (set-cursor (row) col)
      ((null elem)
       (if inc
           (write-line (pack* (car msgs)
             " : caracteristique inconnue")
           )
       )
       (print-elems-of-concept (cdr elems) (cdr types)
                           (cdr msgs) lstasso col inc
       )
      )
      (write-string (pack* (car msgs) " : "))
      (print-elems-of-lst nil elem (row) (column))
      (print-elems-of-concept (cdr elems) (cdr types)
                           (cdr msgs) lstasso col inc
      )
 )
)
((or (equal (car types) 'rdc)
     (equal (car types) 'rdi)
 )
 (let ((elem (cadr (assoc (car elems) lstasso))))
      (set-cursor (row) col)
```

```
          ((null elem)
           (if inc
               (write-line (pack* (car msgs)
                 " : caracteristique inconnue")
               )
           )
           (print-elems-of-concept (cdr elems) (cdr types)
                                   (cdr msgs) lstasso col inc
           )
          )
          (write-string (pack* (car msgs) " : "))
          (print-lsts-of-lst (cadr msgs) elem (row)
                                              (column))
          (print-elems-of-concept (cdr elems) (cdr types)
                                  (cdr msgs) lstasso col inc)
     )
    )
)
(defun acquisition-elems-of-lst (msgs type lst)
    ((null msgs) lst)
    ((equal type 'dc)
     (cons (question-response (pack* (car msgs) " : "))
           (acquisition-elems-of-lst (cdr msgs) type lst)
     )
    )
    ((equal type 'di)
     (cons (string-number
             (question-response (pack* (car msgs) " : "))
           )
           (acquisition-elems-of-lst (cdr msgs) type lst)
     )
    )
)
(defun acquisition-elems-of-elem (msg type lst)
    ((equal type 'rc)
     (let ((elem (question-response (pack* msg " : "))))
          (if (equal elem '"")
              lst
              (cons elem
                    (acquisition-elems-of-elem msg
                                               type lst
              )     )
    ))   )
    ((equal type 'ri)
     (let ((elem (string-number
                   (question-response (pack* msg " : "))))
          (if (equal elem '"")
              lst
              (cons (string-number elem)
                  (acquisition-elems-of-elem msg type lst)
              )
    ))    )
)
```

```
(defun print-elems-of-lst (msgs lst row column)
   ((null lst))
   (set-cursor row column)
   (if (null (car msgs))
       (write-line (string-downcase (car lst)))
       (write-line (pack* (car msgs) " : "
                  (string-downcase (car lst)))
       )
   )
   (print-elems-of-lst (cdr msgs) (cdr lst) (row) column)
)
(defun print-lsts-of-lst (msgs lst row column)
   ((null lst))
   (set-cursor row column)
   (write-line (build-string msgs (car lst)))
   (print-lsts-of-lst msgs (cdr lst) (row) column)
)
(defun print-components-of-mixture (lstcomponents)
   ((null lstcomponents))
   (set-cursor (row) 11)
   (write-line (pack* (caar lstcomponents) "
                      dans une proportion de "
                      (cadar lstcomponents) " %"
   )           )
   (print-components-of-mixture (cdr lstcomponents))
)
(defun question-acquisition-concept ()
   (terpri)
   (o-ou-n  "Voulez-vous ajouter ou modifier des
             caracteristiques ? "
   )
)
(defun acquisition-char (lst read-char rds wrs)
   (terpri 2)
   (write-string
     "Appuyez sur une touche pour continuer "
   )
   (clear-input)
   (write-line (string-downcase (read-char)))
   lst
)
(defun string-number (string i tot dec nbrdec)
   (if (not (numberp i)) (setq i 0))
   (if (not (numberp tot)) (setq tot 0))
   (if (not (numberp nbrdec)) (setq nbrdec 0))
   ((null (char string i)) tot)
   ((eq (char string i) '.)
    (string-number string (add1 i) tot t (+ nbrdec 10))
   )
   (if dec
       (string-number string
                      (add1 i)
                      (+ tot
```

```
                        (* (- (char-code (char string i))
                              (char-code "0")
                           )
                           (/ 1 nbrdec)
                        )
                     )
                     t
                     (+ nbrdec 10)
            )
            (string-number string
                     (add1 i)
                     (+ (* tot 10)
                        (- (char-code (char string i))
                           (char-code "0")
                        )
                     )
                     dec
                     nbrdec
            )
      )
)
(defun question-response (msg)
      (write-string msg)
      (clear-input)
      (string-upcase (read-line))
)
(defun o-ou-n (msg char read-char rds wrs)
      (((null msg))
       (fresh-line)
       (write-string (pack* msg " (o/n) "))
      )
      (clear-input)
      (loop
         (setq char (string-upcase (read-char)))
         ((eq char 'o) (write-line (string-downcase char))
                       t
         )
         ((eq char 'n) (write-line (string-downcase char))
                       nil
         )
         (write-byte 7)
      )
)
```

BASE DE REGLES

```
((ei1 (rule ei1
       (if (substance + (nom (>> contaminant)) + (pi (>> pic)) +)
           (echantillon + (proteine (>> proteine)) +)
           (substance + (nom (<< proteine)) + (pi (>> pi)) +)
           ((ope or) (res t)
            ((ope >=) (res t) (<< pic)
                      ((ope add1) (res nonil) (<< pi))
            )
            ((ope <=) (res t) (<< pic)
                      ((ope sub1) (res nonil) (<< pi))
            )
           )
       )
       (then ((id elimines) (a substances ((<< contaminant)))))
      )
 )
 (ei2 (rule ei2
       (if (substance + (nom (>> contaminant)) + (pi (>> pic)) +)
           (echantillon + (proteine (>> proteine)) +)
           (substance + (nom (<< proteine)) + (pi (>> pi)) +)
           ((ope and) (res t)
            ((ope <) (res t) (<< pic)
                     ((ope add1) (res nonil) (<< pi))
            )
            ((ope >) (res t) (<< pic)
                     ((ope sub1) (res nonil) (<< pi))
            )
           )
       )
       (then ((id contaminants) (a substances ((<< contaminant)))))
      )
 )
 (ei3 (rule ei3
       (if (elimines (substances (>> elimines)))
        (contaminants (substances (>> contaminants)))
           ((ope >=) (res t)
            ((ope length) (res nonil) (<< elimines))
            ((ope sub1) (res nonil)
                        ((ope length) (res nonil) (<< contaminants))
            )
           )
       )
       (then ((id technique) (m nom (ei)))
             ((ope remove-assertion) (res nonil)
              (elimines (substances (<< elimines)))
             )
             ((ope remove-assertion) (res nonil)
              (contaminants (substances (<< contaminants)))
             )
       )
      )
```

```
)
(ei4 (rule ei4
      (if (echantillon + (proteine (>> proteine)) +)
          (substance + (nom (<< proteine)) + (pi (>> pi))
                              (domsta ((>> bi) (>> bs))) +
          )
          ((ope >=) (res t) ((ope abs) (res nonil)
                              ((ope -) (res nonil) (<< bi) (<< pi))
                             )
                             ((ope abs) (res nonil)
                              ((ope -) (res nonil) (<< bs) (<< pi))
                             )
          )
      )
      (then ((id technique) (m ph ((<< bi)))))
     )
)
(ei5 (rule ei5
      (if (echantillon + (proteine (>> proteine)) +)
          (substance + (nom (<< proteine)) + (pi (>> pi))
           (domsta ((>> bi) (>> bs))) +
          )
          ((ope <) (res t) ((ope abs) (res nonil)
                             ((ope -) (res nonil) (<< bi) (<< pi))
                            )
                            ((ope abs) (res nonil)
                             ((ope -) (res nonil) (<< bs) (<< pi))
                            )
          )
      )
      (then ((id technique) (m ph ((<< bs)))))
     )
)
(ei6 (rule ei6
      (if (technique (nom ei) + (ph (>> ph)) +)
          (echantillon + (proteine (>> proteine)) +)
          (substance + (nom (<< proteine)) + (pi (>> pi)) +)
          ((ope >) (res t) (<< ph) (<< pi))
      )
      (then ((id technique) (m gel (deae-sephadex-a50))))
     )
)
(ei7 (rule ei7
      (if (technique (nom ei) + (ph (>> ph)) +)
          (echantillon + (proteine (>> proteine)) +)
          (substance + (nom (<< proteine)) + (pi (>> pi)) +)
          ((ope <) (res t) (<< ph) (<< pi))
          ((ope <=) (res t) (<< ph) 5)
      )
      (then ((id technique) (m gel (sp-sephadex-c50))))
     )
)
(ei8 (rule ei8
      (if (technique (nom ei) + (ph (>> ph)) +)
```

```
            (echantillon + (proteine (>> proteine)) +)
            (substance + (nom (<< proteine)) + (pi (>> pi)) +)
            ((ope <) (res t) (<< ph) (<< pi))
            ((ope >) (res t) (<< ph) 5)
         )
         (then ((id technique) (m gel (cm-sephadex-c50))))
        )
)
(ei9 (rule ei9
        (if (technique (nom ei) + (gel (>> gel)) +)
            (echantillon + (q (>> q)) +)
        )
        (then ((id technique)
               (m volgel (((ope *) (res nonil)
                                   ((ope /) (res nonil) (<< q) 3.75)
                                   54
                           ))
                )
               )
        )
       )
)
(ei10 (rule ei10
        (if (technique (nom ei) + (volgel (>> vol)) +)
            (technique (>+ v))
            ((ope print-tech) (res nonil) (<< v))
            ((ope question-hauteur) (res (>> hauteur)))
        )
        (then ((id technique)
               (m d (((ope diameter)
                          (res nonil) (<< vol) (<< hauteur)
                     )
                    )
               )
               (m h ((<< hauteur)))
              )
        )
       )
)
(ei11 (rule ei11
        (if (technique (nom ei) + (ph (>> ph)) + (d (>> d)) +)
            (echantillon + (proteine (>> proteine)) +)
            (substance + (nom (<< proteine)) + (pi (>> pi)) +)
            (tampon (nom (>> tampon)) (pks (>> pks)))
            ((ope compare) (res (>> bis) (nonil)) >= (<< ph)
             ((ope sub-1) (res nonil) (<< pks))
            )
            ((ope compare) (res (>> bis) (nonil))
             >= (<< pi) (<< bis)
            )
            ((ope compare) (res (>> bss) (nonil)) <= (<< ph)
             ((ope add-1) (res nonil) (<< pks))
            )
            ((ope compare) (res (>> bss) (nonil))
```

```
            <= (<< pi) (<< bss)
          )
        )
        (then ((id technique) (a tampon ((<< tampon)))))
      )
)
(ei12 (rule ei12
        (if (technique (nom ei) + (ph (>> ph)) + (d (>> d)) +)
            (echantillon + (proteine (>> proteine)) +)
            (substance + (nom (<< proteine)) + (pi (>> pi)) +)
            (tampon (nom (>> tampon)) (pks (>> pks)))
            ((ope compare) (res (>> bis) (nonil)) >= (<< ph)
                           ((ope sub-1) (res nonil) (<< pks))
            )
            ((ope compare) (res (>> bis) (nonil))
                          < (<< pi) (<< bis)
            )
            ((ope compare) (res (>> bss) (nonil)) <= (<< ph)
                           ((ope add-1) (res nonil) (<< pks))
            )
            ((ope suite) (res (>> bpks)) (<< pks) (<< bis) (<< bss))
        )
        (then ((id tampon-pi-inf)
               (a tampons (((<< tampon) (<< bpks))))
              )
        )
      )
)
(ei13 (rule ei13
        (if (echantillon + (proteine (>> proteine)) +)
            (substance + (nom (<< proteine)) + (pi (>> pi)) +)
            (tampon-pi-inf (tampons (>> tamps)))
            ((ope nearest) (res (>> tampons))
                           below (<< pi) (<< tamps)
            )
        )
        (then ((ope remove-assertion) (res nonil)
               (tampon-pi-inf (tampons (<< tamps)))
              )
              ((id technique) (a tampon (<< tampons)))
        )
      )
)
(ei14 (rule ei14
        (if (technique (nom ei) + (ph (>> ph)) + (d (>> d)) +)
            (echantillon + (proteine (>> proteine)) +)
            (substance + (nom (<< proteine)) + (pi (>> pi)) +)
            (tampon (nom (>> tampon)) (pks (>> pks)))
            ((ope compare) (res (>> bis) (nonil)) >= (<< ph)
                           ((ope sub-1) (res nonil) (<< pks))
            )
            ((ope compare) (res (>> bss) (nonil)) <= (<< ph)
                           ((ope add-1) (res nonil) (<< pks))
            )
```

```
            ((ope compare) (res (>> bss) (nonil))
                           > (<< pi) (<< bss)
            )
            ((ope suite) (res (>> bpks)) (<< pks) (<< bis) (<< bss))
        )
        (then ((id tampon-pi-sup)
               (a tampons (((<< tampon) (<< bpks))))
              )
        )
      )
)
(ei15 (rule ei15
        (if (echantillon + (proteine (>> proteine)) +)
            (substance + (nom (<< proteine)) + (pi (>> pi)) +)
            (tampon-pi-sup (tampons (>> tamps)))
            ((ope nearest) (res (>> tampons))
                           above (<< pi) (<< tamps)
            )
        )
        (then ((ope remove-assertion) (res nonil)
               (tampon-pi-sup (tampons (<< tamps)))
              )
              ((id technique) (a tampon (<< tampons)))
        )
      )
)
(ei16 (rule ei16
        (if (technique (nom ei) + (tampon (>> tamp)) +)
            (echantillon + (proteine (>> proteine)) +)
            (substance + (nom (<< proteine)) + (pi (>> pi)) +
                       (domsta ((>> bi)(>> bs))) +)
            ((ope >) (res t) (<< pi) (<< bs))
        )
        (then ((id technique) (m elution (force-ionique))))
      )
)
(ei17 (rule ei17
        (if (technique (nom ei) + (tampon (>> tamp)) +)
            (echantillon + (proteine (>> proteine)) +)
            (substance + (nom (<< proteine)) + (pi (>> pi)) +
                         (domsta ((>> bi)(>> bs))) +)
            ((ope <) (res t) (<< pi) (<< bi))
        )
        (then ((id technique) (m elution (force-ionique))))
      )
)
(ei18 (rule ei18
        (if (technique (nom ei) + (tampon (>> tamp)) +)
            (echantillon + (proteine (>> proteine)) +)
            (substance + (nom (<< proteine)) + (pi (>> pi)) +
                       (domsta ((>> bi)(>> bs))) +)
            ((ope <=) (res t) (<< pi) (<< bs))
            ((ope >=) (res t) (<< pi) (<< bi))
        )
```

```
        (then ((id technique) (m elution (force-ionique-et-ph))))
       )
)
(ei19 (rule ei19
       (if (technique (nom ei) + (elution (>> elution)) + )
           ((ope question-decrochage) (res (>> decroch)))
       )
       (then ((id technique) (m decroch ((<< decroch)))))
       )
)
(ei20 (rule ei20
       (if (technique (nom ei) + (volgel (>> volgel)) +
                      (decroch e) +)
       )
       (then ((id echantillon) (m vol (((ope /) (res nonil)
                                        (<< volgel) 2
                                       ))
                                )
             )
       )
       )
)
(ei21 (rule ei21
       (if (technique (nom ei) + (volgel (>> volgel)) +
                      (decroch g) +)
       )
       (then ((id echantillon) (m vol ((<< volgel)))))
       )
)
(ei22 (rule ei22
       (if (substance + (nom (>> contaminant)) + (pi (>> pic)) +)
           (echantillon + (proteine (>> proteine)) +)
           (substance + (nom (<< proteine)) + (pi (>> pi)) +)
           (echantillon + (contaminants (+ ((<< contaminant)
                          (>> prop)) +))
           )
       )
       (then ((id echantillon) (m contaminants (nil)))
             (contaminant (<< contaminant) (<< pic) (<< prop))
       )
       )
)
(ei23 (rule ei23
       (if (contaminant (>> contaminant) (>> pic) (>> prop))
           (echantillon + (proteine (>> proteine)) +)
           (substance + (nom (<< proteine)) + (pi (>> pi)) +)
           ((ope >=) (res t) (<< pic)
                     ((ope sub1) (res nonil) (<< pi))
           )
           ((ope <=) (res t) (<< pic) (<< pi))
           ((ope /) (res (>> nprop)) ((ope *) (res nonil)
                                      (<< prop)
                                      ((ope -) (res nonil)
                                       100
```

```
                                        ((ope droite) (res nonil)
                                         ((ope sub1) (res nonil)
                                                     (<< pi)
                                         )
                                         0 (<< pi) 100 (<< pic)
                                        )
                                       )
                                      )
                                      100
            )
        )
        (then ((ope remove-assertion) (res nonil)
               (contaminant (<< contaminant) (<< pic) (<< prop))
              )
              (element de contamination (<< contaminant) (<< nprop))
        )
       )
)
(ei24 (rule ei24
       (if (contaminant (>> contaminant) (>> pic) (>> prop))
           (echantillon + (proteine (>> proteine)) +)
           (substance + (nom (<< proteine)) + (pi (>> pi)) +)
           ((ope >=) (res t) (<< pic) (<< pi))
           ((ope <=) (res t) (<< pic)
                     ((ope add1) (res nonil) (<< pi))
           )
           ((ope /) (res (>> nprop)) ((ope *) (res nonil)
                                      (<< prop)
                                      ((ope -) (res nonil)
                                       100
                                       ((ope droite) (res nonil)
                                                     (<< pi) 100
                                                     ((ope add1)
                                                      (res nonil)
                                                      (<< pi)
                                                     )
                                                     0 (<< pic)
                                       )
                                      )
                                     )
                                     100
            )
        )
        (then ((ope remove-assertion) (res nonil)
               (contaminant (<< contaminant) (<< pic) (<< prop))
              )
              (element de contamination (<< contaminant) (<< nprop))
        )
       )
)
(ei25 (rule ei25
       (if (contaminant (>> contaminant) (>> pic) (>> prop))
           (echantillon + (proteine (>> proteine)) +)
           (substance + (nom (<< proteine)) + (pi (>> pi)) +)
```

```
            ((ope or) (res t)
             ((ope <) (res t) (<< pic)
                      ((ope sub1) (res nonil) (<< pi))
             )
             ((ope >) (res t) (<< pic)
                      ((ope add1) (res nonil) (<< pi))
             )
            )
        )
        (then ((ope remove-assertion) (res nonil)
               (contaminant (<< contaminant) (<< pic) (<< prop))
              )
        )
       )
)
(ei26 (rule ei26
        (if (element de contamination (>> contaminant) (>> prop))
            ((ope zerop) (res nil) (<< prop))
        )
        (then ((ope remove-assertion) (res nonil)
               (element de contamination (<< contaminant) (<< prop))
              )
              ((id echantillon) (a contaminants (((<< contaminant)
                                 (<< prop))))
              )
        )
       )
)
(pr1 (rule pr1
      (if (echantillon + (>> proteine) +)
          (substance + (<< proteine) + (dompre ((>> bi)
                       (>> bs))) +)
          ((ope <=) (res t) (<< bs) 60)
      )
      (then ((id technique) (nom pr)))
     )
)
(pr2 (rule pr2
      (if (echantillon + (>> vol) + (>> proteine) +)
          (substance + (<< proteine) + (dompre ((>> bi)
                       (>> bs))) +)
          ((ope *) (res (>> volpr))
                   ((ope *) (res nonil) (<< vol)
                            ((ope /) (res nonil)
                                     100
                                     ((ope -) (res nonil)
                                              100 (<< bs)
                                     )
                            )
                   )
                   ((ope /) (res nonil) (<< bs) 100)
          )
      )
      (then ((id technique) (volagprec (<< volpr))))
```

```
      )
)
(pr3 (rule pr3
      (if (substance + (nom (>> contaminant)) +
                       (dompre ((>> bic) (>> bsc))) +)
          (echantillon + (contaminants (+ ((<< contaminant)
                          (>> prop)) +)))
      )
      (then ((id echantillon) (m contaminants (nil)))
            (contaminant (<< contaminant) (<< bic) (<< bsc)
                         (<< prop))
      )
     )
)
(pr4 (rule pr4
      (if (contaminant (>> contaminant) (>> bic)
                       (>> bsc) (>> prop))
          (echantillon + (proteine (>> proteine)) +)
          (substance + (nom (<< proteine)) + (dompre ((>> bi)
                       (>> bs))) +)
          ((ope <=) (res t) (<< bs) (<< bic))
      )
      (then ((ope remove-assertion) (res nonil)
             (contaminant (<< contaminant) (<< bic)
                          (<< bsc) (<< prop))
            )
            ((id echantillon)
             (a contaminants (((<< contaminant) (<< prop)))))
      )
     )
)
(pr5 (rule pr5
      (if (contaminant (>> contaminant) (>> bic)
                       (>> bsc) (>> prop))
          (echantillon + (proteine (>> proteine)) +)
          (substance + (nom (<< proteine)) +
                       (dompre ((>> bi) (>> bs))) +)
          ((ope >=) (res t) (<< bs) (<< bsc))
      )
      (then ((ope remove-assertion) (res nonil)
             (contaminant (<< contaminant) (<< bic)
                          (<< bsc) (<< prop))
            )
      )
     )
)
(pr6 (rule pr6
      (if (contaminant (>> contaminant) (>> bic)
                       (>> bsc) (>> prop))
          (echantillon + (proteine (>> proteine)) +)
          (substance + (nom (<< proteine)) +
                       (dompre ((>> bi) (>> bs))) +)
          ((ope /) (res (>> nprop)) ((ope *) (res nonil)
                                     (<< prop)
```

```
                                        ((ope -) (res nonil)
                                         100
                                         ((ope droite) (res nonil)
                                                       (<< bic)
                                                       100
                                                       (<< bsc)
                                                       0
                                                       (<< bs)
                                         )
                                        )
                                       )
                                       100
                )
          )
          (then ((ope remove-assertion) (res nonil)
                  (contaminant (<< contaminant) (<< bic)
                               (<< bsc) (<< prop))
                )
                ((id echantillon)
                 (a contaminants (((<< contaminant) (<< nprop)))))
          )
         )
    )
   )
```

BASE DE META-REGLES

```
((rule mpr1
       (if (pr envisageable)
           ((rule) (pr1 (>> pr1)))
       )
       (then ((ope remove-assertion) (res nonil)
              (pr envisageable)
             )
             ((ope use-rule) (res t) (<< pr1))
       )
 )
 (rule mpr2
       (if (technique (nom pr))
           ((rule) (pr2 (>> pr2)))
           ((rule) (pr3 (>> pr3)))
           ((rule) (pr4 (>> pr4)))
           ((rule) (pr5 (>> pr5)))
           ((rule) (pr6 (>> pr6)))
       )
       (then ((ope use-rule) (res t) (<< pr2))
             ((ope use-rule) (res t) (<< pr3))
             ((ope use-rule) (res t) (<< pr4))
             ((ope use-rule) (res t) (<< pr5))
             ((ope use-rule) (res t) (<< pr6))
             (fin application pr)
       )
 )
 (rule mpr3
       (if (fin application pr)
           ((ope remove-assertion) (res nonil) (fin application pr))
           (echantillon (>+ echantillon))
           (technique (>+ technique))
           ((ope print-tech) (res nonil) (<< technique))
           ((ope print-echantillon) (res t) (<< echantillon))
       )
       (then (pr accepte))
 )
 (rule mpr4
       (if (pr accepte)
           (technique (>+ caracteristiques))
       )
       (then ((ope remove-assertion) (res nonil)
              (technique (<+ caracteritiques))
             )
             ((ope remove-assertion) (res nonil) (pr accepte))
             (ei envisageable)
             (tm envisageable)
             (melange non dissout)
       )
 )
 (rule mpr5
       (if (technique (nom pr) (>+ caracteristiques)))
       (then ((ope remove-assertion) (res nonil)
```

```
                (technique (nom pr) (<+ caracteristiques))
              )
        )
)
(rule mei1
      (if (ei envisageable)
          ((rule) (ei1 (>> ei1)))
          ((rule) (ei2 (>> ei2)))
          ((rule) (ei3 (>> ei3)))
      )
      (then ((ope remove-assertion) (res nonil) (ei envisageable))
            ((ope use-rule) (res t) (<< ei1))
            ((ope use-rule) (res t) (<< ei2))
            ((ope use-rule) (res t) (<< ei3))
      )
)
(rule mei2
      (if (technique (nom ei))
          ((rule) (ei3b (>> ei3b)))
          ((rule) (ei4 (>> ei4))) ((rule) (ei5 (>> ei5)))
          ((rule) (ei6 (>> ei6))) ((rule) (ei7 (>> ei7)))
          ((rule) (ei8 (>> ei8))) ((rule) (ei9 (>> ei9)))
          ((rule) (ei10 (>> ei10))) ((rule) (ei11 (>> ei11)))
          ((rule) (ei12 (>> ei12))) ((rule) (ei13 (>> ei13)))
          ((rule) (ei14 (>> ei14))) ((rule) (ei15 (>> ei15)))
          ((rule) (ei16 (>> ei16))) ((rule) (ei17 (>> ei17)))
          ((rule) (ei18 (>> ei18))) ((rule) (ei19 (>> ei19)))
          ((rule) (ei20 (>> ei20))) ((rule) (ei21 (>> ei21)))
          ((rule) (ei22 (>> ei22))) ((rule) (ei23 (>> ei23)))
          ((rule) (ei24 (>> ei24))) ((rule) (ei25 (>> ei25)))
          ((rule) (ei26 (>> ei26)))
      )
      (then ((ope use-rule) (res t) (<< ei3b))
            ((ope use-rule) (res t) (<< ei4))
            ((ope use-rule) (res t) (<< ei5))
            ((ope use-rule) (res t) (<< ei6))
            ((ope use-rule) (res t) (<< ei7))
            ((ope use-rule) (res t) (<< ei8))
            ((ope use-rule) (res t) (<< ei9))
            ((ope use-rule) (res t) (<< ei10))
            ((ope use-rule) (res t) (<< ei11))
            ((ope use-rule) (res t) (<< ei12))
            ((ope use-rule) (res t) (<< ei13))
            ((ope use-rule) (res t) (<< ei14))
            ((ope use-rule) (res t) (<< ei15))
            ((ope use-rule) (res t) (<< ei16))
            ((ope use-rule) (res t) (<< ei17))
            ((ope use-rule) (res t) (<< ei18))
            ((ope use-rule) (res t) (<< ei19))
            ((ope use-rule) (res t) (<< ei20))
            ((ope use-rule) (res t) (<< ei21))
            ((ope use-rule) (res t) (<< ei22))
            ((ope use-rule) (res t) (<< ei23))
            ((ope use-rule) (res t) (<< ei24))
```

```
                ((ope use-rule) (res t) (<< ei25))
                ((ope use-rule) (res t) (<< ei26))
                (fin application ei)
        )
    )
    (rule mei3
          (if (fin application ei)
              ((ope remove-assertion) (res nonil) (fin application ei))
              (echantillon (>+ echantillon))
              (technique (>+ technique))
              ((ope print-tech) (res nonil) (<< technique))
              ((ope print-echantillon) (res t) (<< echantillon))
          )
          (then (ei accepte))
    )
    (rule mei4
          (if (ei accepte)
              (technique (>+ caracteristiques))
          )
          (then ((ope remove-assertion) (res nonil)
                 (technique (<+ caracteritiques))
                )
                ((ope remove-assertion) (res nonil) (ei accepte))
                (ei envisageable)
                (tm envisageable)
                (pr envisegeable)
          )
    )
    (rule mei5
          (if (technique (nom ei) (>+ caracteristiques)))
          (then ((ope remove-assertion) (res nonil)
                 (technique (nom ei) (<+ caracteristiques))
                )
          )
    )
)
```

BASE DE FAITS

```
(
 (colonne ((technique (tm))
           (diametre 0.5)
           (hauteur 80)
           (vitesse 2)
           (volume 0.25)
          )
          ((technique (tm))
           (diametre 1)
           (hauteur 80)
           (vitesse nil)
           (volume 1)
          )
          ((technique (tm))
           (diametre 2)
           (hauteur 80)
           (vitesse 15)
           (volume 3.5)
          )
          ((technique (tm))
           (diametre 4)
           (hauteur 80)
           (vitesse 40)
           (volume 20)
          )
 )

 (gel ((technique (tm))
       (type Sephadex-G-75)
       (marque Pharmacia)
       (domfrac (6000 65000))
       (nonsep 1.5)
      )
      ((technique (tm))
       (type Ultrogel-ACA-44)
       (marque Lkd)
       (domfrac (10000 120000))
       (nonsep 1.5)
      )
      ((technique (tm))
       (type Sephacryl-S200)
       (marque Pharmacia)
       (domfra (15000 160000))
       (nonsep 2)
      )
      ((technique (tm))
       (type Sepharose-4B)
       (marque Pharmacia)
       (domfrac (100000 nil))
       (nonsep 3)
```

```
      )
)

(tampon ((nom phosphate) (pks (7.2)))
        ((nom tris) (pks (8.2)))
        ((nom citrate) (pks (3.1 4.7 5.4)))
        ((nom acide-acetique) (pks (4.75)))
        ((nom cacodilate) (pks (6.2)))
        ((nom glycine) (pks (9.9)))
)

(melange ((nom X)
          (substances
           ((aa 12) (bb 10) (cc 15))
          )
         )
)

(substance ((nom aa)
            (nature enzyme)
            (pm 35000)
            (pi 6)
            (domsta (4 8))
            (dompre (35 55))
            (duree-sta 15)
           )
           ((nom bb)
            (nature enzyme)
            (pm 60000)
            (pi 4.7)
            (domsta (4 6))
            (dompre (65 85))
            (duree-sta 4)
           )
           ((nom cc)
            (nature proteine)
            (pm 90000)
            (pi 5)
            (domsta (5 9))
            (dompre (25 50))
            (duree-sta 12)
           )
)
)
```